PROJECT REPORT ON

“CONTAINMENT ZONE ALERTING APPLICATION”

SUBMITTED BY

RAKESH L	312319104125
KASPROV ARVIN	312319104066
KAVINKANT RAJA	312319104067
KARTHIKEYAN	312319104065

NOVEMBER ,2022

ST.JOSEPH’S COLLEGE OF ENGINEERING

OMR CHENNAI - 600119

TABLE OF CONTENT

1.INTRODUCTION

An Android application that can inform people about the Covid-19 containment zones and prevent them from entering them. This Android app updates the locations of the areas in a Google map that have been identified as containment zones**.** The app also alerts users when they enter a containment zone and uploads the user's information to an online database. Many Google tools and APIs, such as Firebase and Geofencing API, are used in this application to achieve all of these functionalities. As a result, this application can be used to raise further social awareness about the need for precautionary measures to be taken by the people**.**

1.2 PURPOSE

We focus on developing a mobile-based application to provide information about the Covid-19 containment zones in in this paper. The application also tracks the user's location and sends an alert if the user enters a containment zone. The application also provides users with daily Covid-19 case survey data to keep them up to date. The application is built with the Android SDK and stores location data in the Firebase Cloud Firestore. The geofencing client for Android is used to create geofences around containment zones, and the notification manager is used to provide notifications.Users can view the location of the containment zones via the Android application. It also alerts the user when he or she crosses the boundary of a containment zone or remains within it.

2.LITERATURE SURVEY

S.No	TITLE	PURPOSED WORK	APPARATUS/ ALGORITHM	TECHNOLOGY	MERITS & DEMERITS
1	Development of an Android Application for Viewing Covid-19 Containment Zones and Monitoring Violators Who are Trespassing into it using Firebase and Geofencing.	The primary goals of this project are to notify users whenever they enter a containment zone, update the position of the area on a Google map, alert users when they do, and upload the user's IMEI number to an online database..	1. Geofencing API 2. Firebase API 3. Location Tracking 4. IMEI Number 5. Andriod SDK	Cloud Technology	This application can be used as a tool for creating further social awareness about the arising need of precautionary to be taken by the people of India
2	Application for Covid-19 Real Time Counter.	Efficient way of showing the identified Covid-19 containment zone.Futher more lie maritine and forest safety to prevent user from entering restricted areas.	• Time Series Analysis. • Location Tracking	• Cloud Technology	The Application can include various government organization to help act faster.

S.NO	TITLE	PROPOSED WORK	TOOLS USED/ ALGORITHM	TECHNOLOGY	ADVANTAGES/ DISADVANTAGES
3	Aarogya Setu (COVID-19).	In this study, a methodology is given for displaying current Covid statistics..	• Bluetooth • GPS • Digital ID	• Cloud Technology	At a time user can see Covid-19 total cases, active cases and discharge cases.
4	Tracking the Covid Zone through Geo-fencing technique. - JULY 10 2020	In order to track the Covid Zones and improve and tighten security measures, this study will provide a methodology..	• Bluetooth Based Application. • DRDO Netra. • National Intelligence Grid(NATGRID).	• Mobile Network • Cloud Technology	The major issue with those Bluetooth based application is that tracking can be done only if the enabled the Bluetooth option.

S.No	TITLE	PROPOSED WORK	TOOLS USED/ ALGORITHM	TECHNOLOGY	ADVANTAGES/ DISADVANTAGES
5	A Compact Wearable - IOT(W-IOT) system for Health Safety and Protection of Outgoers in the Post-Lockdown World(COVID-19 Lifeguard).	The primary emphasis of this work is an IOT-based health monitoring system that uses a variety of sensors to measure bodily parameters and issues alerts in case of emergencies..	• Spo2 detector. • GPS • Bluetooth Module • GSM Modem	• IOT Technology	The body temperature, heart rate, and oxygen saturation levels have to be monitored regularly.
6	Evaluating how smartphone contact tracing technology can reduce the spread of infectious diseases(COVID-19).	This essay argues that managing epidemics depends on being able to identify and stop the spread of infectious disorders like Covid-19.	• Epidemic Model • Web Scraping • Opportunistic Network(OPP NET) • Web scrapping	• Mobile Computing	Accurate technologies such as Bluetooth allow for greater selectivity when it comes to quarantining people.

2.1 EXISTING PROBLEM

According to the survey, several apps have been developed in the country to battle and contain COVID-19. Most states in our country have their own apps with specific features and functionality to assist their citizens in stopping the spread of COVID-19, obtaining medical assistance during a crisis, raising awareness, and understanding safety precautions.

2.2 REFERENCES

[1]. COVID-19 outbreak: Migration, effects on society, global environment and prevention, https://www.sciencedirect.com/science/article/pii/S0048969720323998

[2]. BBC News, “WHO head: ‘Our key message is: test, test, test.’” https://www.bbc.co.uk/news/av/world51916707/whohead-our-key-message-is-test-test- test.

[3]. Pethick, “Developing antibody tests for SARS-CoV-2,” Lancet, vol. 395 (10230).

[4]. E. Hernández-Orallo, P. Manzoni, C. T. Calafate and J. Cano, "Evaluating How Smartphone Contact Tracing Technology Can Reduce the Spread of

Infectious Diseases: The Case of COVID-19," in IEEE Access, vol. 8, pp. 99083-99097, 2020, doi: 10.1109/ACCESS.2020.2998042.

[5]. How Reliable and Effective Are the Mobile Apps Being Used to Fight COVID-19?, https://thewire.in/tech/COVID-19- mobile-apps-india

[6]. A flood of coronavirus apps are tracking us. Now it's time to keep track of them, https://www.technologyreview.com/2020/05/07/1000961/lau nching-mittr-COVID-19- tracing-tracker/

[7]. Now, a mobile app predicts COVID-19 incidence days in advance, https://www.thehindu.com/sci-tech/science/now-amobile-app-predicts-COVID-19-incidence-days-inadvance/article31544706.ece

[8]. COVID-19 apps around the world, https://techerati.com/features-hub/opinions/COVID-19- apps-around-the-world/

[9]. 5G mobiles do not spread COVID-19, https://www.who.int/emergencies/diseases/novelcoronavirus-2019/advice-for-public/mythbusters?gclid=Cj0KCQjwudb3BRC9ARIsAEavUvdlwAV59nTqxfJ1xp7nKMD9TZsiT4mksqnq11xrTuO3 7kL9m1qwwaAj_tEALw_wc B#5g

[10]. Coronavirus Apps: Every App the Central Government And States Have Deployed to Track COVID-19

[11]. https://gadgets.ndtv.com/apps/features/central-stategovernments-launch-coronavirus- mobile-app-list-2204286

[12]. Corona virus app API, https://coronavirus.app/map?compared=US,DE,FR,GB,ES

[13]. Mizoram launches coronavirus app to disseminate authoritative info to the public https://www.deccanchronicle.com/technology/in-othernews/040420/mizoram-launches-coronavirus-app-todisseminate-authoritative-info-to.html

[14]. Aarogya Setu- Features and tools https://en.wikipedia.org/wiki/Aarogya_Setu

[15]. Top-10 smartphone apps to track COVID19,https://www.geospatialworld.net/blogs/popular-appsCOVID-19

[16]. COVID-19 Apps. https://en.wikipedia.org/wiki/COVID19_apps

[17]. COVID-19: The world embraces contact-tracing technology to fight the virus https://www.livemint.com/news/world/COVID-19-theworld-embraces-contact-tracing-

[18]. CheckCOVID-19Now: A web app to spot coronavirus cases in Telangana, https://www.newindianexpress.com/cities/hyderabad/2020/apr/18/checkCOVID-19now-a-web-app-to-spot-coronaviruscases-in-telangana-2131600.html

[19]. Geo-Fence ,Applications, https://en.wikipedia.org/wiki/Geofence

[20]. TOMTOM Geo Fencing API documentation, https://developer.tomtom.com/geofencing- api/geofencingapi-documentation

[21]. Firebase in-app messaging, https://firebase.google.com/docs/in-app-messaging/composecampaign?authuser=2

[22]. Sending messages to multiple devices, https://firebase.google.com/docs/cloudmessaging/android/send-multiple?authuser=2

[23]. Firebase console database https://firebase.google.com/docs/database/usage/monitorusage?authuser=2

2.3 PROBLEM STATEMENT

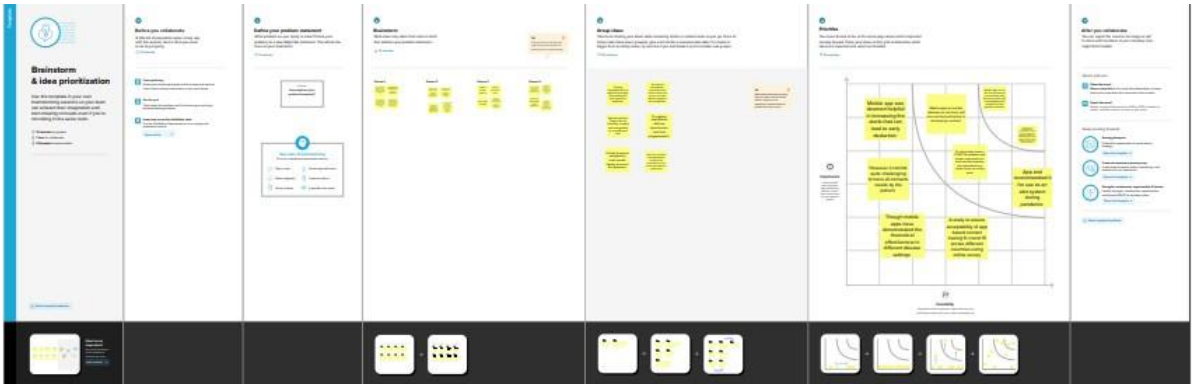
- Many users have found this app buggy and had reported with login issues.
- COVID Symptom Tracker App also falls short of being incredibly helpful to scientists and data analysts.
- This apps seems inadequate to find out symptoms in the patients who were affected by Coronavirus earlier and recovered later.

3.IDEATION & PROPOSED SOLUTION

Ideation is the process where you generate ideas and solutions through sessions such as Sketching, Prototyping, Brainstorming, Brainwriting, Worst Possible Idea, and a wealth of other ideation techniques. Ideation is also the third stage in the Design Thinking process. Proposed Solution means the technical solution to be provided by the Implementation agency in response to the requirements and the objectives of the Project.

3.1 Empathy Map Canvas

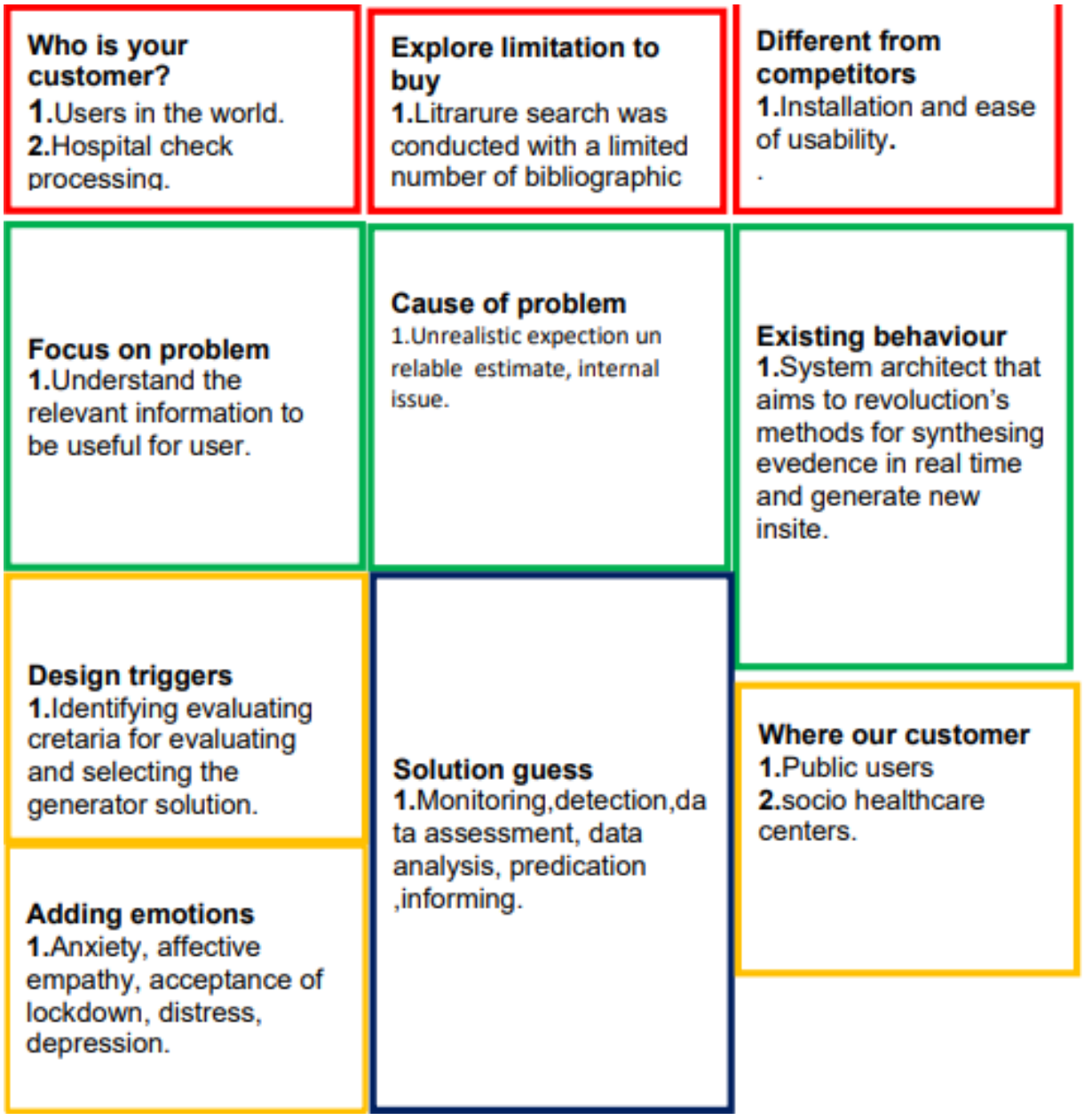
3.2 IDEATION & BRAINSTROMING



3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Statistics data from RESTful API.Data snapshot from firebase for new data or data snapshot from cache. Containment zones shown on a google map and covid 19 statistics on a bottom sheet. User receives notification on entering a containment zone.
2.	Idea / Solution description	1.Retrieving diagnosis 2.Retrieving Exposure configuration 3.Segmentation 4.Protocal documentation 5.Classification 6.Contributing
3.	Novelty / Uniqueness	1.Android application updates location of areas which are identified to be the containment zone. 2.Application further extract the IMEI number of the trespasser and upload to the online data base. 3.The application prompt background location service permission and if granted the geo fences get trigged even in application is not open in foreground.
4.	Social Impact / Customer Satisfaction	1..Medical Drone deliveries 2. Situational awareness lockdown curfew enforcement. 3.Broadcasting useful information.
5.	Business Model (Revenue Model)	
6.	Scalability of the Solution	The produce the vaccination scenarios and quantity the under reporting impact.Record identification Databases(n=380)Other source(n=10).Duplicate removed before record moved for reasons(n=14).

3.4 PROBLEM SOLUTION FIT



4.REQUIREMENT ANALYSIS

4.1 FUCTIONAL REQUIREMENTS

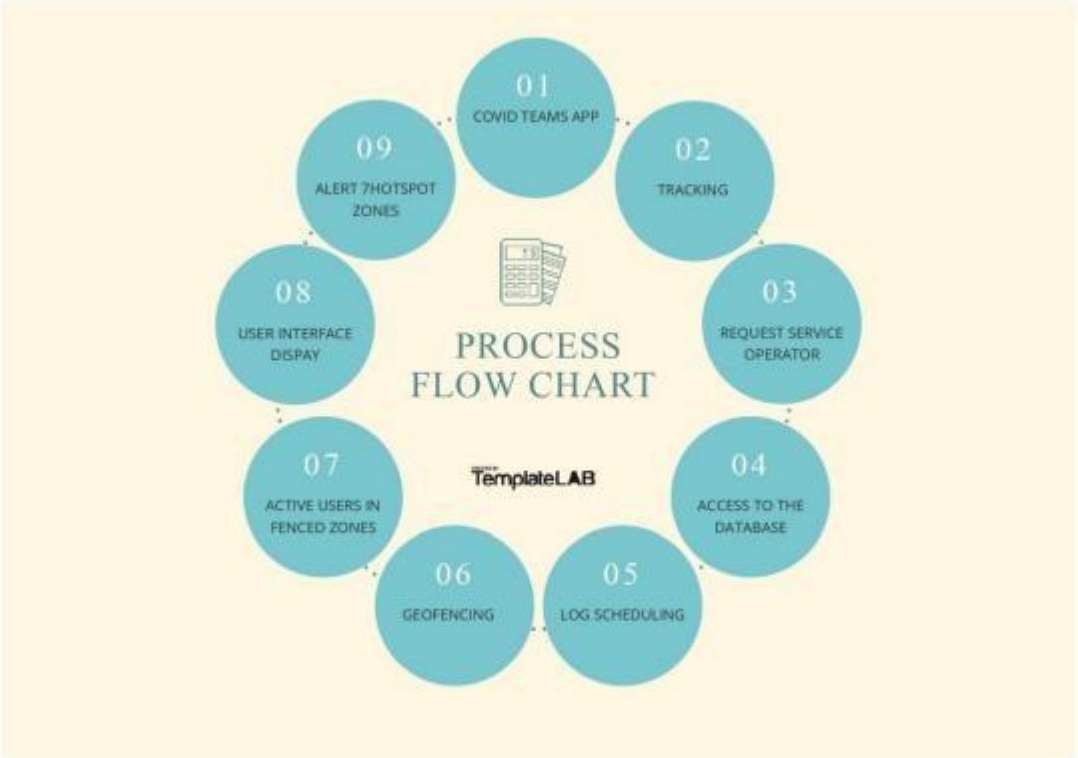
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Users can sign up using their email address or existing phone number.
FR-2	User Confirmation	Confirmation can be completed by sending a verification code via email or using an OTP.
FR-3	Track the location	Utilizing the Google map API, track the intruders and update the areas marked as containment zones on the Google map.
FR-4	Affected areas are shown	Geo fence serves as a warning for trespassers, and containment zones were labeled using zone colours.
FR-5	Alert notification	If the user enters the containment zone, a notification or message will be sent by tracking their location using GPS.

4.2 NON-FUNCTIONAL REQUIREMENTS

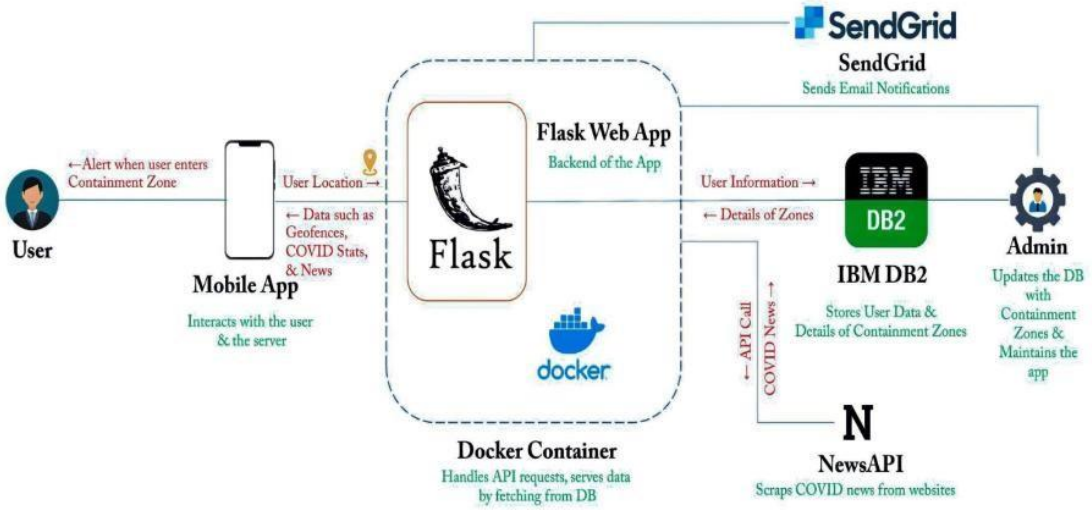
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The COVID-19 investigative process is more effective and thorough because to the user interface, which is particularly easy to use when compared to other interfaces.
NFR-2	**Security**	The user's data will be secured.
NFR-3	**Reliability**	The user may travel securely and rely on the information provided by the programme..
NFR-4	**Performance**	The Geofencing and GPS technologies can be used to produce the most suitable results.
NFR-5	**Scalability**	It is possible to access this application from anywhere, and the zone information is accurate.

5.PROJECT DESIGN

5.1 DATAFLOW DIAGRAM



5.2 SOLUTION & TECHNICAL ARCHITECTURE



5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (covid team app)	Registration	USN-1	As a user, I can register for the application by entering my email address, password, and password confirmation..	I can access my account/dashboard	High	Sprint-1
		USN-2	As a user, I will receive a confirmation email once I have registered for the application.	I can receive a confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook.	I can register & access the dashboard with Facebook Login	Low	Sprint-4
		USN-4	As a user, I can register for the application through Gmail.	I can register & access the dashboard with Google Login	Medium	Sprint-1
		USN-5	As a user, I can register for the application through Twitter.	I can register & access the dashboard with Twitter Login	Low	Sprint 4
	Login	USN-6	As a user, I can log into the application by entering my email & password	I can access it whenever I want its access.	High	Sprint-1
	Dashboard	USN-7	As a user, I need to give permission to access My Contacts, Location, and Storage.	I get access to their services	High	Sprint-2
		USN-8	As a user, I get access to the dashboard which shows a map with marked zones	I can see the zone information on the dashboard.	high	Sprint-2
Hospitals Administrator	Registration	USN-9	As a management, I need to register my hospitals on the site.	I can see the registered hospital in the hospital dashboard.	high	Sprint-1
	Login	USN-10	As a management, I need to login into my dashboard with my given hospital id and password.	I can see my dashboard after login.	medium	Sprint-1
	Dashboard	USN-11	As a management, I need to enter the case information of the patient that visits our hospital.	I can view the patient information on the dashboard.	high	Sprint-2
		USN-12	As a management, I need to store all the patient information on the cloud	-	high	Sprint-3
Administrator	Services	USN-13	As an admin, I need to provide valid information about the pandemic out there.	I can get the pandemic updates out there.	high	Sprint-2
		USN-14	As an admin, I need to provide medical advice through a chatbot.	I get medicinal recommendations through a chatbot.	medium	Sprint-3
		USN-15	As an admin, I need to provide medical recommendations by collaborating with top hospitals.	I get medical instruction through chief doctors.	low	Sprint-3
		USN-16	As an admin, I need to alert the user when they enter pandemic zones.	I got a notification when I am in the pandemic area.	Medium	Sprint-4
		USN-17	As an admin, I need to provide preventive measures when they travel through it.	I got a remedies notification when I am in the pandemic area.	high	Sprint-3
		USN18	As an admin, I need to provide special services for premium users by giving services like monitoring health by their smart bands.	I was treated special after becoming a premium member.	low	Sprint 4
	Data collections	USN-18	As an admin, I need to store all the user information on the cloud	I can access my information when I needed	Medium	Sprint-4

6.PROJECT PLANNING & SCHEDULING

'Project planning' is fundamentally about choosing and developing effective policies and methodologies to achieve project goals. While 'project scheduling' is a procedure for assigning tasks and completing them by allocating appropriate resources within an estimated budget and time frame.

6.1 SPRINT PLANNING & ESTIMATION

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the selected project & gathering information by referring the, technical papers,research publications etc.	19 OCTOBER 2022
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	18 OCTOBER 2022
Ideation	List the by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	18 OCTOBER 2022
Proposed Solution	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	18 OCTOBER 2022
Problem Solution Fit	Prepare problem - solution fit document.	18 OCTOBER 2022

Solution Architecture	Prepare solution architecture document.	15 OCTOBER 2022
Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	24 OCTOBER 2022
Functional Requirement	Prepare the functional requirement document.	22 OCTOBER 2022
Data Flow Diagrams	Draw the data flow diagrams and submit for review.	26 OCTOBER 2022
Technology Architecture	Prepare the technology architecture diagram.	22 OCTOBER 2022
Prepare Milestone & Activity List	Prepare the milestones & activity list of the project.	31 OCTOBER 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	28 OCTOBER 2022

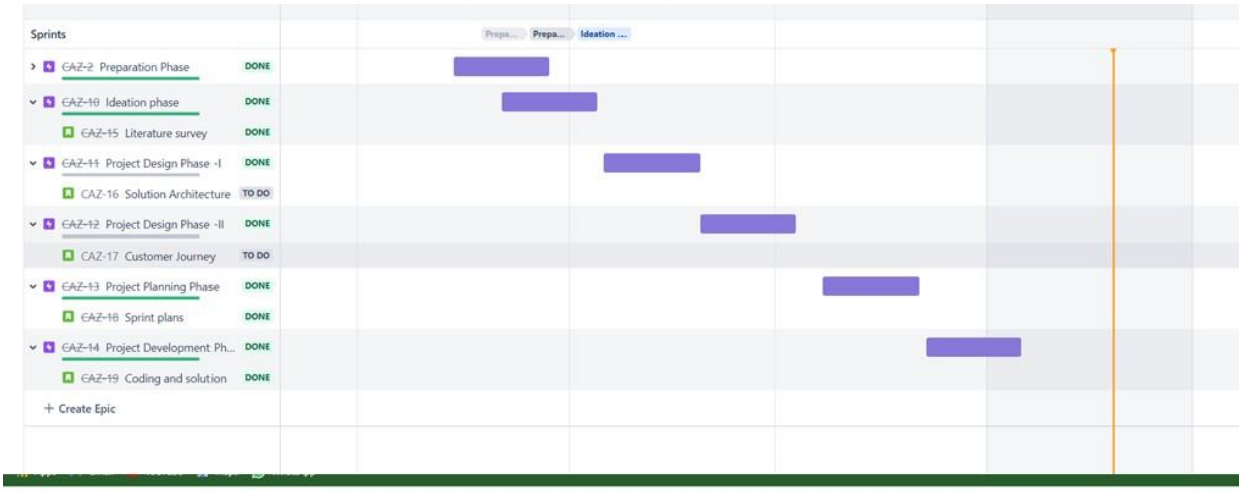
6.2 SPRINT DELIVERY SCHEDULE

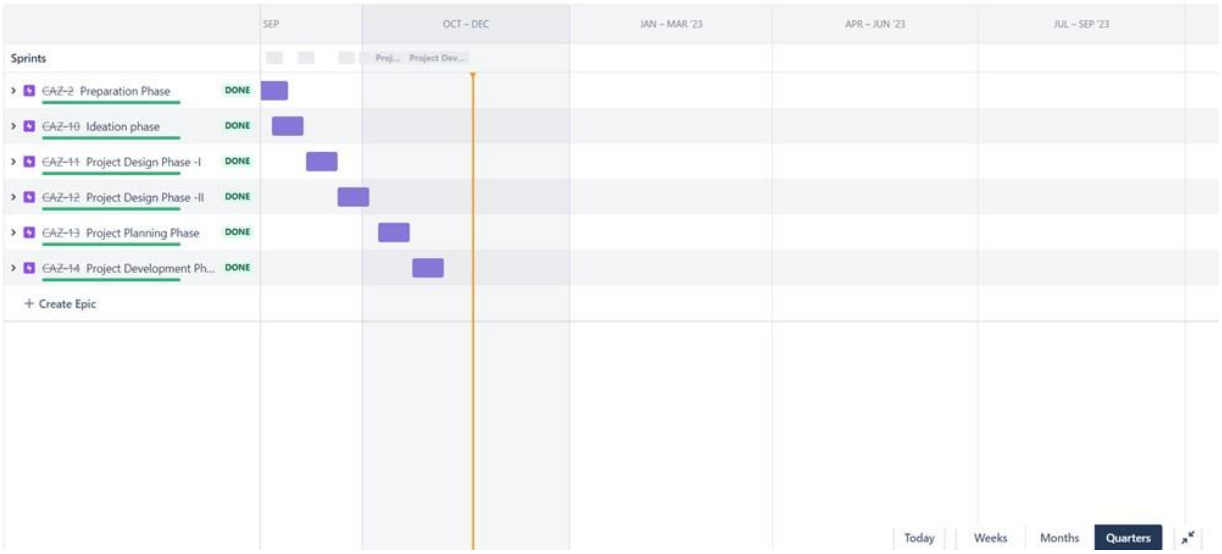
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	User: I can register for the application by entering my email, password and verifying password.	3	High	Ruchitha
		USN-2	User: I will receive a confirmation email once I have registered for the application.	2	High	Rubasri
		USN-3	User: I can register for the application through Gmail.	5	Medium	Preethi
		USN-4	Management: I need to register my hospitals on the site.	2	High	Gopika
		USN-5	User: I can log into the application by entering my email & password	3	High	Ruchitha
	Login	USN-6	Management: I need to login into my dashboard with my given hospital id and password.	5	Medium	Preethi
	Dashboard	USN-7	User: I need to give permission to access my Contacts, Location, and Storage	5	High	Ruchitha

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2		USN-8	User: I get access to the dashboard which shows a map with containment zones	5	High	Gopika
		USN-9	Management: I need to enter the case information of the patient that visits our hospital.	5	High	Rubasri
	Services	USN-10	Admin: I need to provide valid information about the pandemic out there.	5	High	Preethi
Sprint-3	Dashboard	USN-11	Management: I need to store all the patient information on the cloud.	5	High	Ruchitha
	Services	USN-12	Admin: I need to provide medical advice through a chatbot.	5	Medium	Gopika
		USN-13	Admin: I need to provide medical recommendations by collaborating with top hospitals.	5	Low	Rubasri
		USN-14	Admin: I need to provide preventive measures when they travel through it.	5	High	Preethi
	Registration	USN-15	User: I can register for the application through Facebook.	2	Low	Gopika
Sprint-4		USN-16	User: I can register for the application through Twitter.	2	Low	Rubasri
	Services	USN-17	Admin: I need to alert the user when they enter pandemic zones.	3	Medium	Ruchitha
		USN-18	Admin: I need to provide special services for premium users by giving services like monitoring health by their smart bands.	3	Low	Gopika
	Data Collection	USN-19	Admin: I need to store all the user information on the cloud	5	Medium	Ruchitha

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
		USN-20	Admin: I need to collect the recent list of diseases in the world.	5	Low	Preethi

6.3 REPORTS FROM JIRA





7.CODING & SOLUTIONING

7.1 FEATURE 1

To track the Covid zones, to enhance and tighten the security measures. A firebase is created for the containment zone. The person who enters or exits out of that particular zone will be monitored and alert message will be sent to that person's mobile.

Add permissions in firebase

This permission allows the application to connect to the internet and save data.

<uses-permission android:name="android.permission.INTERNET"/>

Add the Google Maps location dependency.

implementation 'com.google.android.gms:play-services-location:17.0.0'

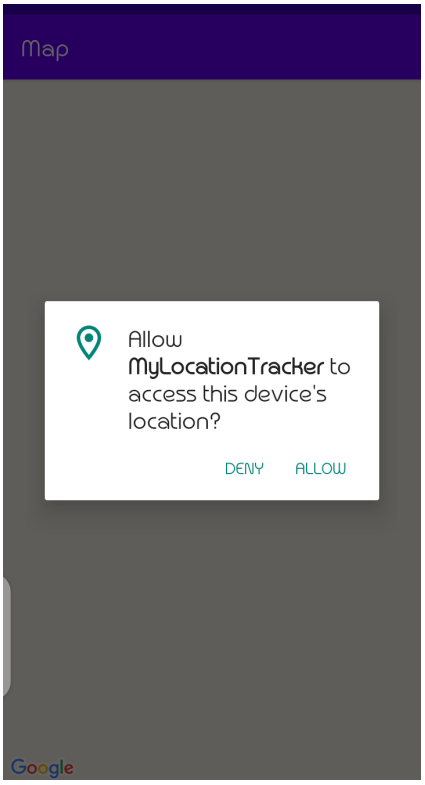
MapsActivity.kt

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback {

    private lateinit var map: GoogleMap

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_maps)
        // Obtain the SupportMapFragment and get notified when the map is ready
to be used.
        val mapFragment = supportFragmentManager
                .findFragmentById(R.id.map) as SupportMapFragment
        mapFragment.getMapAsync(this)
        setupLocClient()

    }

    private lateinit var fusedLocClient: FusedLocationProviderClient
    // use it to request location updates and get the latest location

    override fun onMapReady(googleMap: GoogleMap) {
        map = googleMap //initialise map
```

```
        getCurrentLocation()
    }
    private fun setupLocClient() {
        fusedLocClient =
            LocationServices.getFusedLocationProviderClient(this)
    }

    // prompt the user to grant/deny access
    private fun requestLocPermissions() {
        ActivityCompat.requestPermissions(this,
            arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),
//permission in the manifest
            REQUEST_LOCATION)
    }

    companion object {
        private const val REQUEST_LOCATION = 1 //request code to identify
specific permission request
        private const val TAG = "MapsActivity" // for debugging
    }

    private fun getCurrentLocation() {
        // Check if the ACCESS_FINE_LOCATION permission was granted
before requesting a location
        if (ActivityCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION) !=
            PackageManager.PERMISSION_GRANTED) {

          // call requestLocPermissions() if permission isn't granted
            requestLocPermissions()
        } else {

            fusedLocClient.lastLocation.addOnCompleteListener {
                // lastLocation is a task running in the background
                val location = it.result //obtain location
                //reference to the database
                val database: FirebaseDatabase = FirebaseDatabase.getInstance()
                val ref: DatabaseReference = database.getReference("test")
                if (location != null) {

                    val latLng = LatLng(location.latitude, location.longitude)
                   // create a marker at the exact location
                    map.addMarker(MarkerOptions().position(latLng)
```

```
                    .title("You are currently here!"))
                // create an object that will specify how the camera will be updated
                val update = CameraUpdateFactory.newLatLngZoom(latLng,
16.0f)

                map.moveCamera(update)
                //Save the location data to the database
                ref.setValue(location)
            } else {
                // if location is null , log an error message
                Log.e(TAG, "No location found")
            }



        }
    }
}


override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<String>,
    grantResults: IntArray) {
    //check if the request code matches the REQUEST_LOCATION
    if (requestCode == REQUEST_LOCATION)
    {
        //check if grantResults contains PERMISSION_GRANTED.If it does,
call getCurrentLocation()
        if (grantResults.size == 1 && grantResults[0] ==
            PackageManager.PERMISSION_GRANTED) {
            getCurrentLocation()
        } else {
            //if it doesn`t log an error message
            Log.e(TAG, "Location permission has been denied")
        }
```

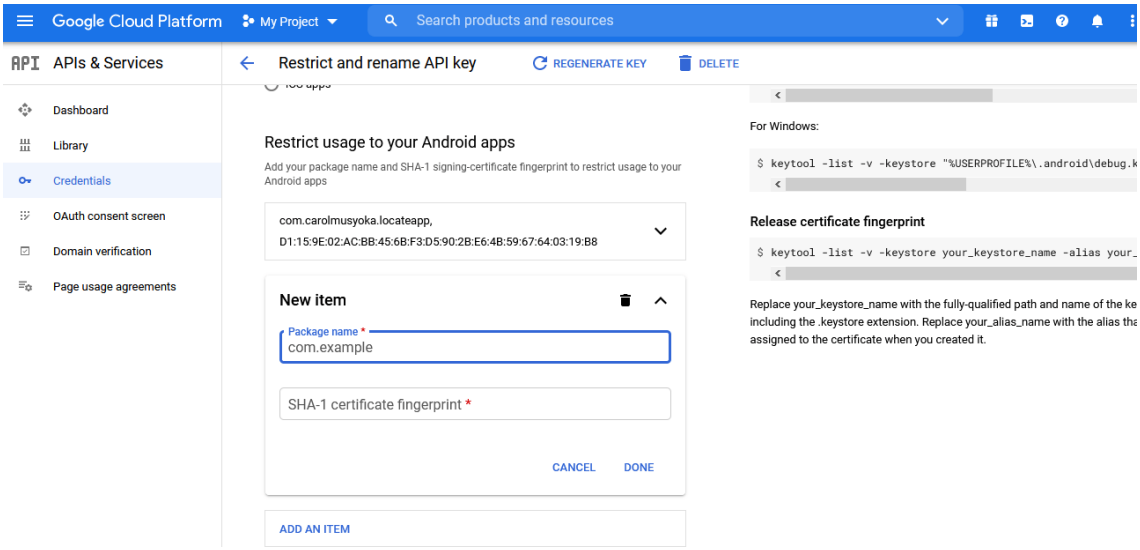
RUN THE APP


Map
Allow MyLocationTracker to access this device's location?
DENY
ALLOW
Google

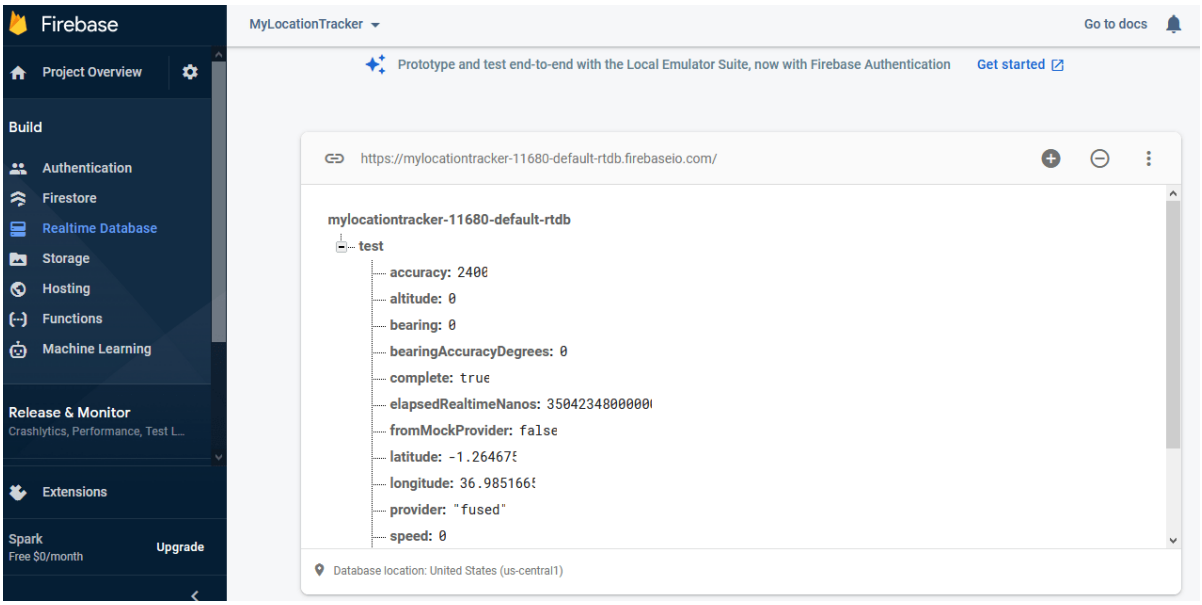

Map
bis Ruai
By Faith Shop
Prime Auto Solutions
You are currently here!
Kangundo Rd
Precious Life
Medical Centre
Flexx Mart
Absa | Atm | Ruai
Google

FEATURE-2

User uploads locally stored tracked locations and home address after testing positive for COVID-19. These locations are stored in cloud storage. Note that, no other user information except the locations are accessible by other users.



DATABASE CONNECTION



7.3 DATABASE SCHEMA

8. TESTING

Testing is finding out how well something works**.** Testing in Software Engineering is defined as an activity to check whether the actual results match the expected results.

8.1 TEST CASES

A test case includes information such as test steps, expected results and data while a test scenario only includes the functionality to be tested.

1. Login button click with wrong credentials entered.
2. Signup with already registered mail ID.
3. Signup with wrong form data entered.
4. Entering home page with logged out session.
5. Clicking home page buttons with logged out session.
6. Invalid data entered in change password page and requested for change in password.

8.2 USER ACCEPTANCE TESTING

User Acceptance Testing (UAT) is a process to check whether the system accepts a user's requirements.

User acceptance testing is the final testing stage in software development before production. It's used to get feedback from users who test the software and its user interface (UI). UAT is usually done manually, with users creating real-worldsituations and testing how the software reacts and performs.

S.NO	TEST CASE	REQUIRED OUTPUT	RESULT OUTPUT	STATUS
1	Login button click with wrong credentials	Wrong credentials entered notification	Wrong credentials entered notification	ACCEPTED
2	Signup with already registered mail ID.	Email already registered notification	Email already registered notification	ACCEPTED
3	Signup with wrong form data entered.	Wrong credentials entered notification	Wrong credentials entered notification	ACCEPTED
4	Entering home page with logged out session.	Take user to login page	Take user to login page	ACCEPTED
5	Clicking home page buttons with logged out session.	Take user to login page	Take user to login page	ACCEPTED
6	Invalid data entered in change password page and requested for change in password.	Wrong form data entered notification	Wrong form data entered notification	ACCEPTED

Scenario	Test Step	Expected Result	Actual Outcome
Verify that the input field that can accept maximum of 10 characters	Login to application and key in 10 characters	Application should be able to accept all 10 characters.	Application accepts all 10 characters.
Verify that the input field that can accept maximum of 11 characters	Login to application and key in 11 characters	Application should NOT accept all 11 characters.	Application accepts all 10 characters.

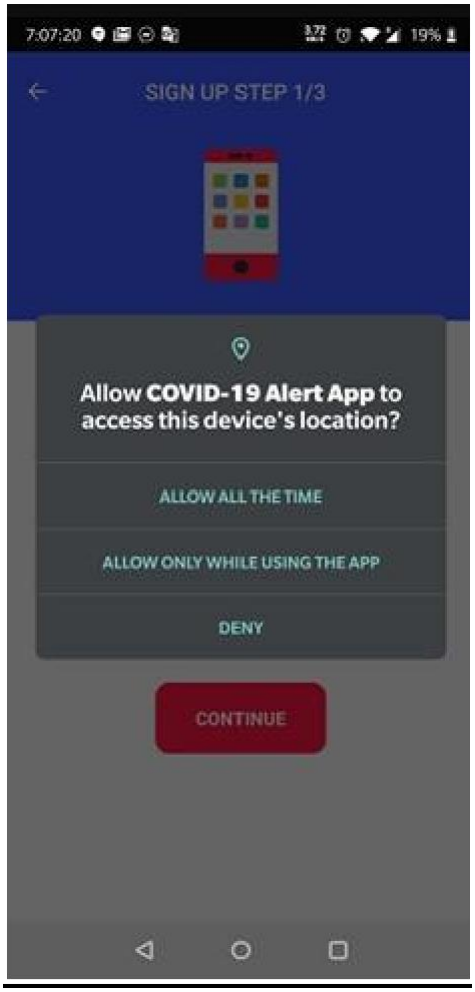
9.RESULT

Result is an ideal result that the tester should get after the test case is performed. It's usually documented together with the test case. It's usually compared with actual result, and if the actual result differs from the expected one, the difference is documented and called a bug.

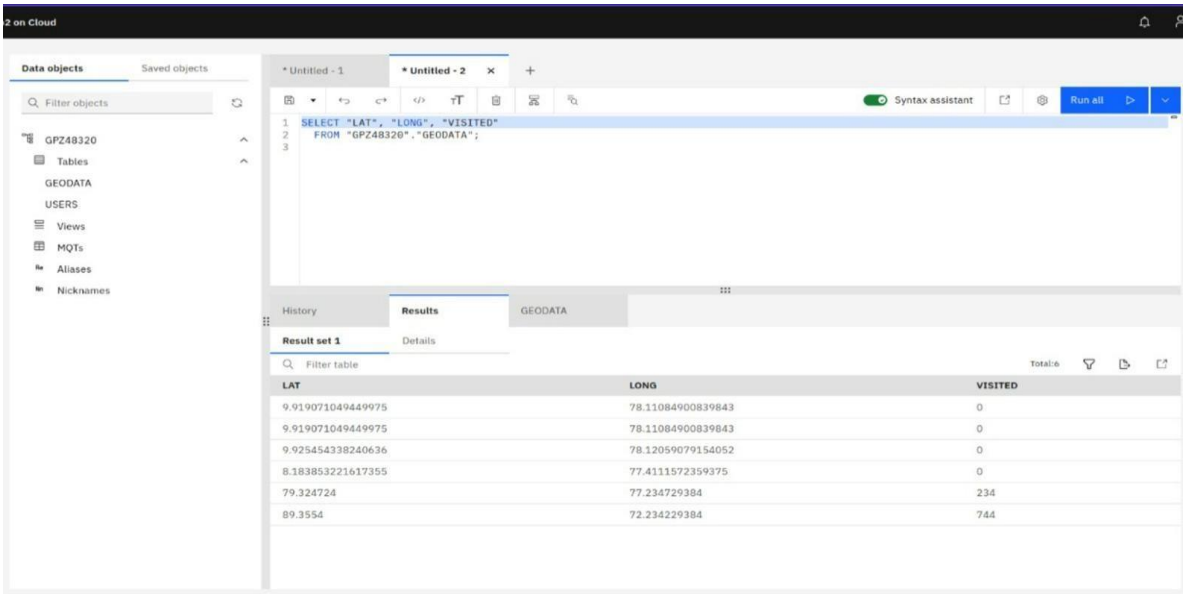
Infected-locations

map-boxed-area

Permission for location access

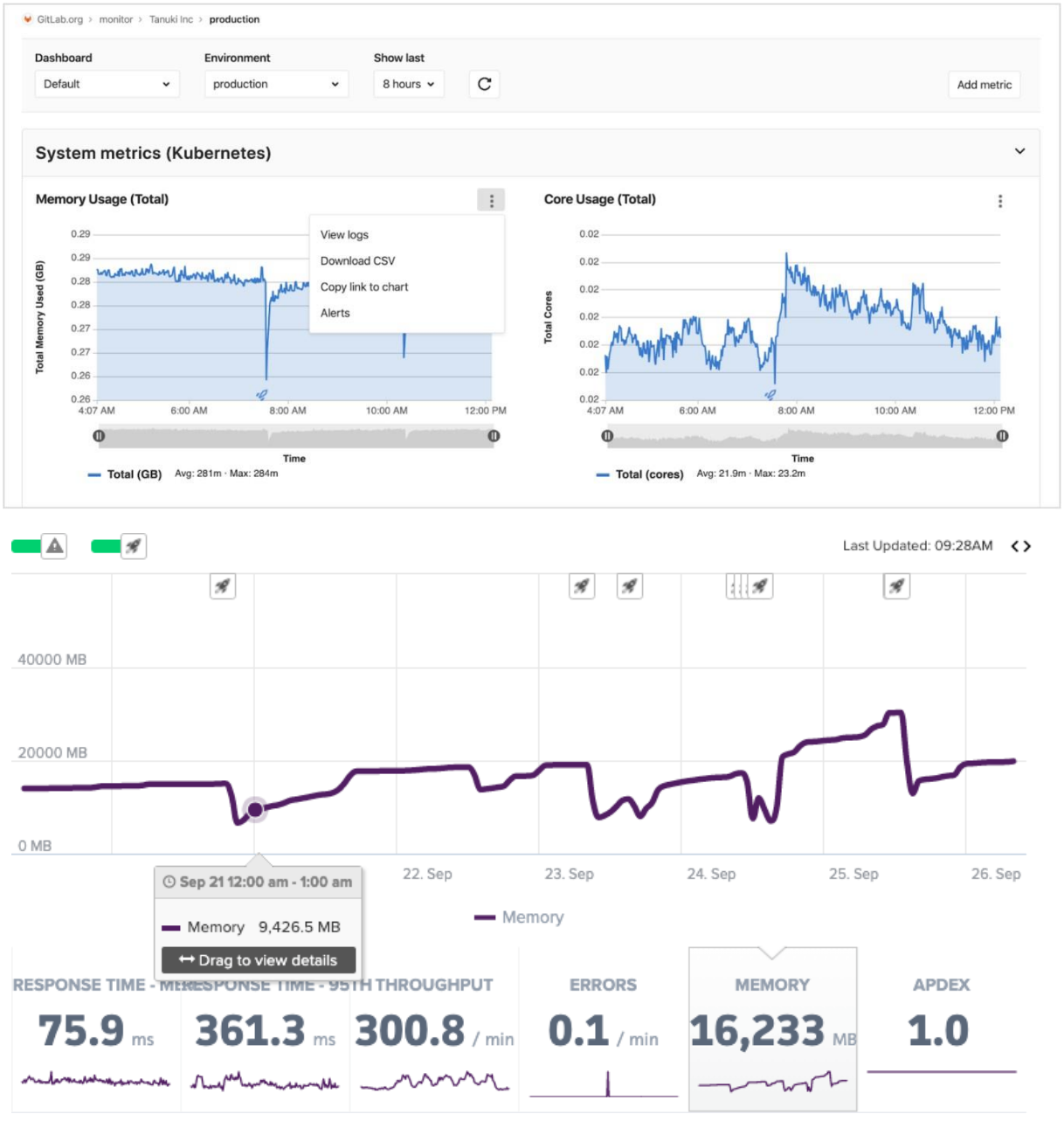


DB CONNECTION



9.1 PERFORMANCE METRICES

The performance metrics definition refers to the measurement of behavior, activities, and overall performance of a business**.**



10. ADAVANTAGE & DISADVANTAGE

ADVANTAGES

- Users can know if they have been near a person suspected to be affected by COVID-19.
- It sends separate notification alerts to the user on entering.
- It is the easiest tool to predict COVID-19 Contaminant zones.
- Users will receive a real-time notification whenever they are in the same location as an infected user.
- Exposure notifications are delivered more effectively if the process is automatic. It makes sense – rather than having to call each person individually, an app may alert everyone that's been in close proximity to an infected person.

DISADVANTAGES

There are some privacy concerns including: lack in privacy

- Access to personal information and geographical location.
- User's data exposure.
- Mostly regarding the centralisation vs. decentralisation of data. For the most part, though, academics agree that moving away from location-based contact tracing apps is essential for user privacy.

11. CONCLUSION

This app is intended to alert people about containment zones in a specific region by continuously monitoring an individual's location. The application is developed on Android SDK and uses Firebase Cloud Firestore to store the location data. Android's geofencing client is used to create geofences the containment zones and notification manager is used to provide notifications. The application's key benefits include monitoring people's activity and alerting them to their safety movements. Through the app's global news feed, relief requests can be posted without directly sharing personal or family information of a user. A contact button is attached to relief posts through which any other user can call and contact the relief request post's author and reach out for help. This feature especially targets the middle-class families that are suffering greatly in silence and cannot seek help publicly. A user is allowed to make only

one relief post every seven days, this is a measure taken to stop misuse of the feature.

12. FUTURE SCOPE

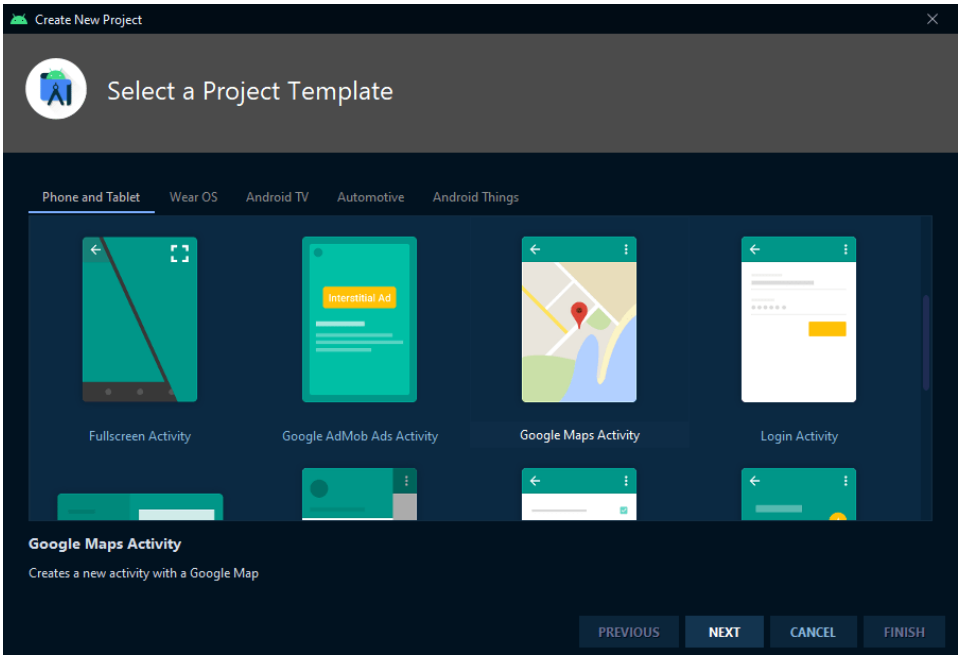
The application efficiently displays the identified Covid-19 containment zones to users in a Google map. With the alarming increase in Covid-19 affected cases around the world, this developed application can be used to raise social awareness among the general public. This application also monitors the user's location and determines whether it is in the list of identified containment zones. On entering, it sends the user separate notification alerts. The developed Android application also extracts the trespasser's email address in the containment zones, which can be used by to track and identify people who frequently trespass the containment zones. As a result, this application identifies the containment zones and emphasises the importance of taking additional precautionary measures to combat Covid-19. The application has been tested in several locations and has proven to produce accurate result.

9.APPENDIX

Creating a Location-tracking App using Firebase and Google Maps in Android.

Step 1 - Creating a new project

Using thr google map template

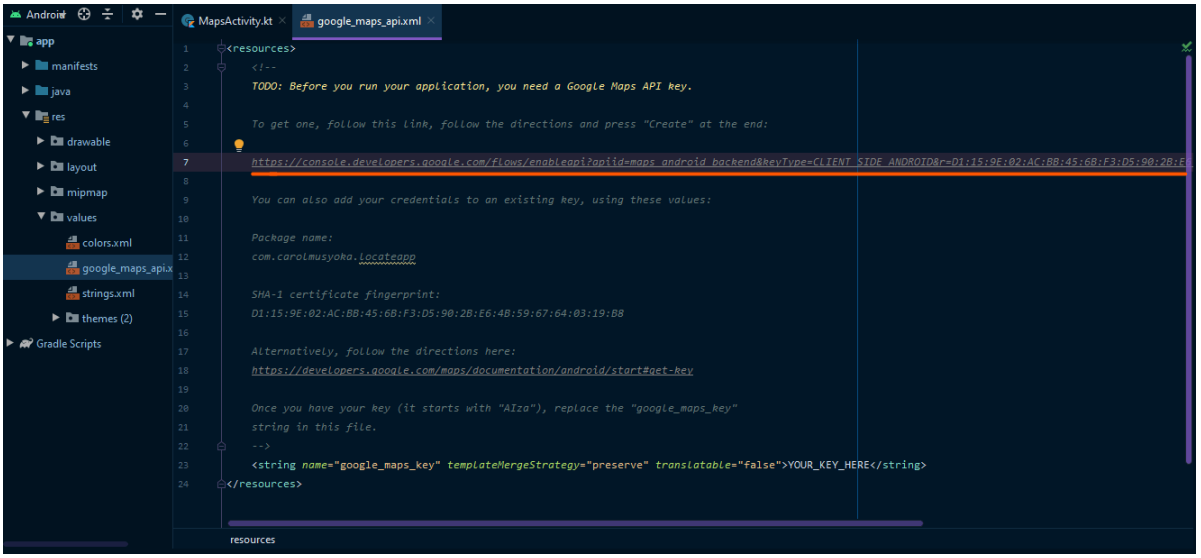


com.google.android.geo.API_KEY specifies the API key.

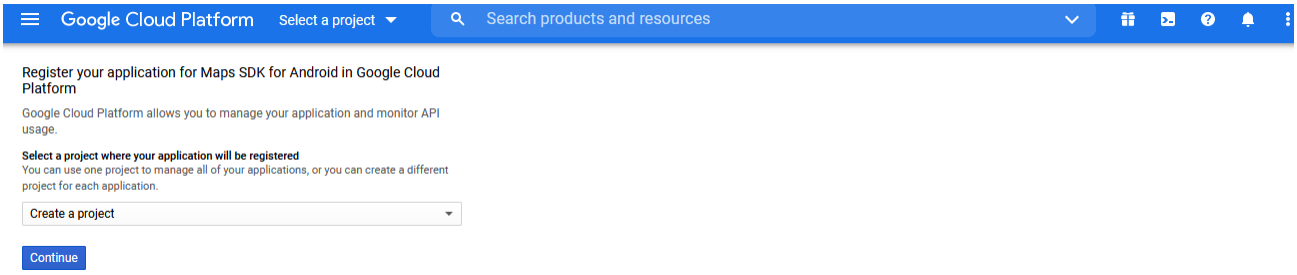
implementation **'com.google.android.gms:play-services-maps:17.0.0'**

Step-2 Create an API key

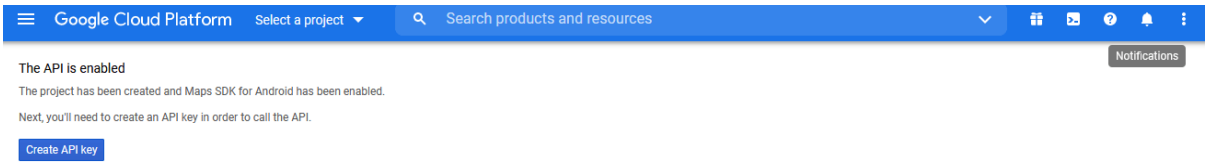
Open res/values/google_maps_api.xml.

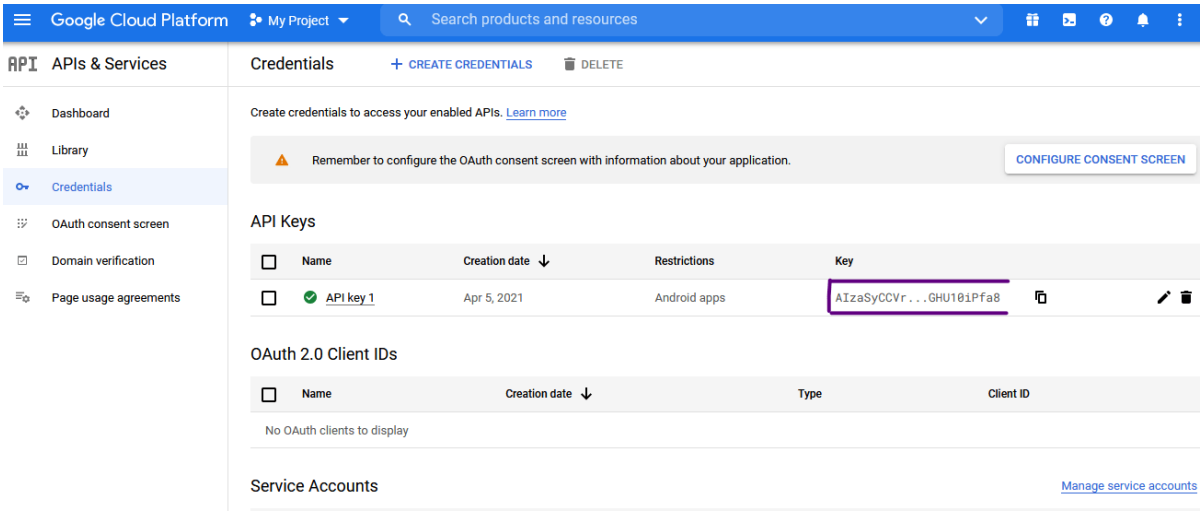


This file will contain your API key

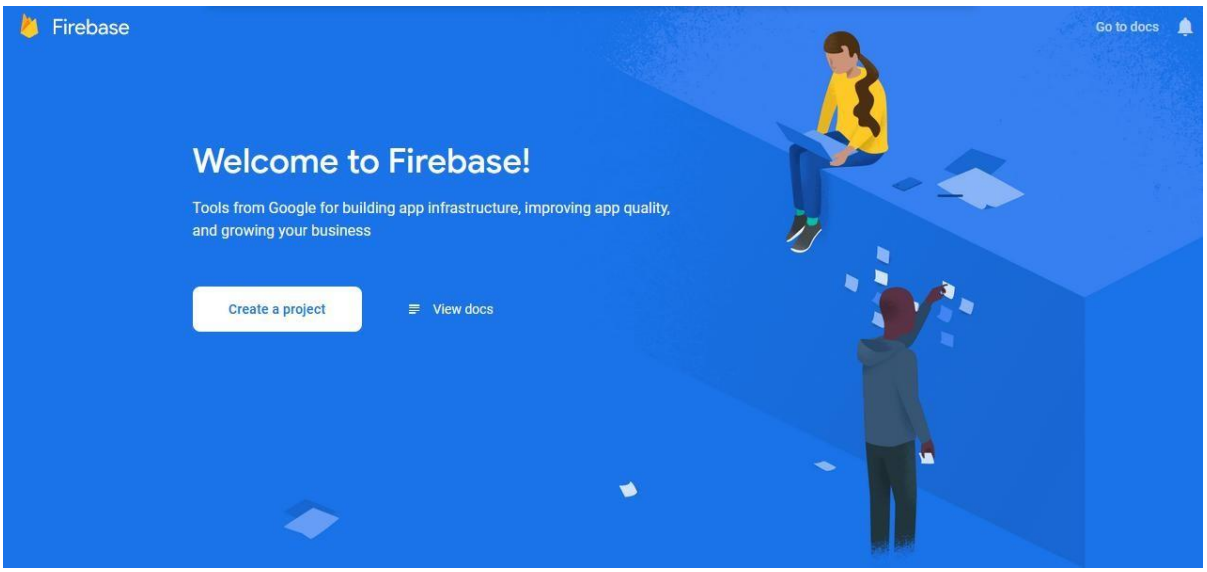


API key to call the API.



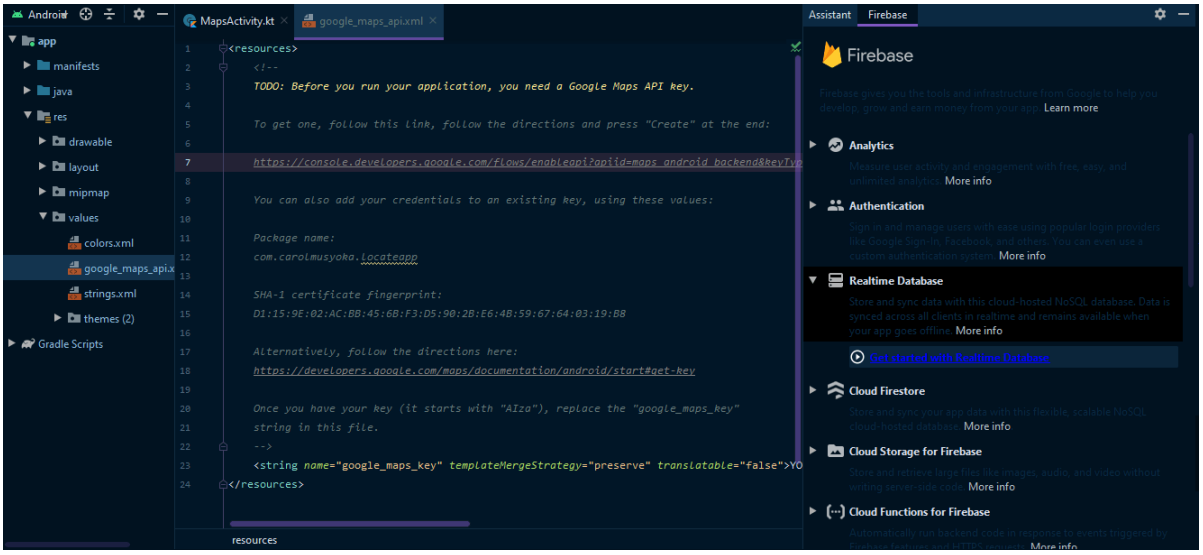


Step 3 - Creating a Firebase project



Step 4 - Connect the Firebase project to the app

- Go to tools>firebase



Step 5 - Add permissions

- Add the internet permission.

This permission allows the application to connect to the internet and save data.

<uses-permission android:name="android.permission.INTERNET"/>

- Add the Google Maps location dependency.

  implementation 'com.google.android.gms:play-services-location:17.0.0'

Step 6 - The MapsActivity

Navigate to MapsActivity.kt

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback {

    private lateinit var map: GoogleMap

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_maps)
        // Obtain the SupportMapFragment and get notified when the map is ready
to be used.
```

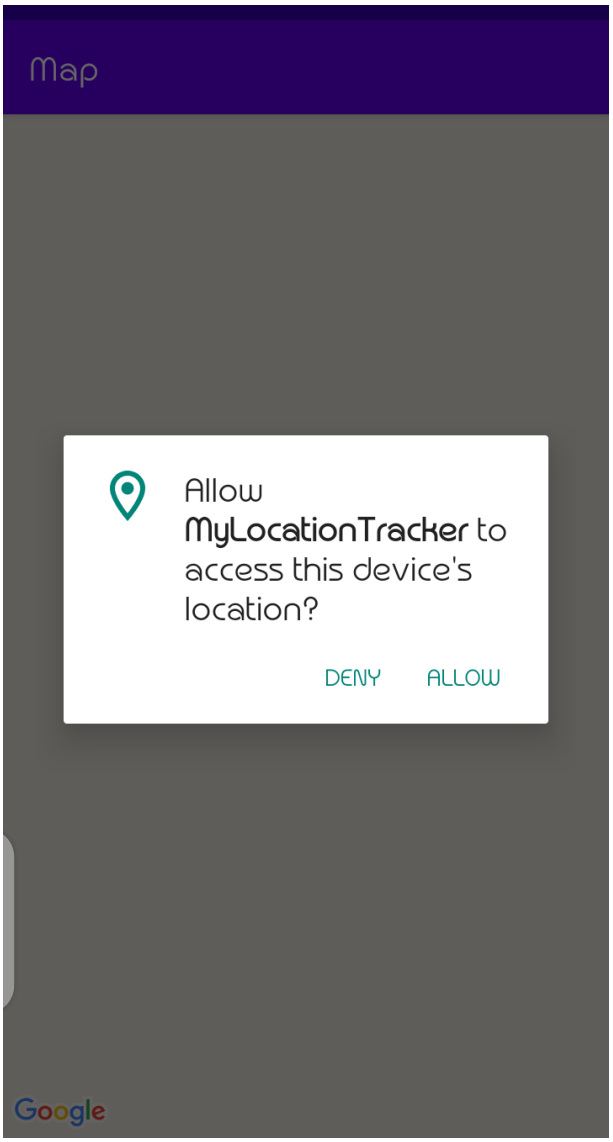
```
        val mapFragment = supportFragmentManager
                .findFragmentById(R.id.map) as SupportMapFragment
        mapFragment.getMapAsync(this)
        setupLocClient()

    }

    private lateinit var fusedLocClient: FusedLocationProviderClient
    // use it to request location updates and get the latest location

    override fun onMapReady(googleMap: GoogleMap) {
        map = googleMap //initialise map
        getCurrentLocation()
    }
    private fun setupLocClient() {
        fusedLocClient =
            LocationServices.getFusedLocationProviderClient(this)
    }

    // prompt the user to grant/deny access
    private fun requestLocPermissions() {
        ActivityCompat.requestPermissions(this,
            arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),
//permission in the manifest
            REQUEST_LOCATION)
    }

    companion object {
        private const val REQUEST_LOCATION = 1 //request code to identify
specific permission request
        private const val TAG = "MapsActivity" // for debugging
    }

    private fun getCurrentLocation() {
        // Check if the ACCESS_FINE_LOCATION permission was granted
before requesting a location
        if (ActivityCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION) !=
            PackageManager.PERMISSION_GRANTED) {

          // call requestLocPermissions() if permission isn't granted
            requestLocPermissions()
        } else {
```

```
            fusedLocClient.lastLocation.addOnCompleteListener {
                // lastLocation is a task running in the background
                val location = it.result //obtain location
                //reference to the database
                val database: FirebaseDatabase = FirebaseDatabase.getInstance()
                val ref: DatabaseReference = database.getReference("test")
                if (location != null) {

                    val latLng = LatLng(location.latitude, location.longitude)
                   // create a marker at the exact location
                    map.addMarker(MarkerOptions().position(latLng)
                        .title("You are currently here!"))
                    // create an object that will specify how the camera will be updated
                    val update = CameraUpdateFactory.newLatLngZoom(latLng,
16.0f)

                    map.moveCamera(update)
                    //Save the location data to the database
                    ref.setValue(location)
                } else {
                     // if location is null , log an error message
                    Log.e(TAG, "No location found")
                }



            }
        }
    }


    override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<String>,
        grantResults: IntArray) {
        //check if the request code matches the REQUEST_LOCATION
        if (requestCode == REQUEST_LOCATION)
        {
            //check if grantResults contains PERMISSION_GRANTED.If it does,
call getCurrentLocation()
            if (grantResults.size == 1 && grantResults[0] ==
                PackageManager.PERMISSION_GRANTED) {
```

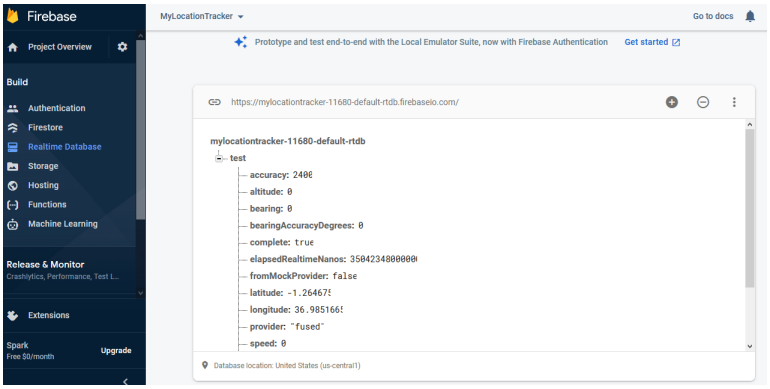
```
            getCurrentLocation()
        } else {
            //if it doesn`t log an error message
            Log.e(TAG, "Location permission has been denied")
        }
    }
  }
}
```

Step 7 - Run the app

Run the app. This is will achieve (locations may differ). Give the app location permission.



Successfully saved the user's location in a database. Navigate to the Firebase console and click on the project had created.



The LocationChecker app

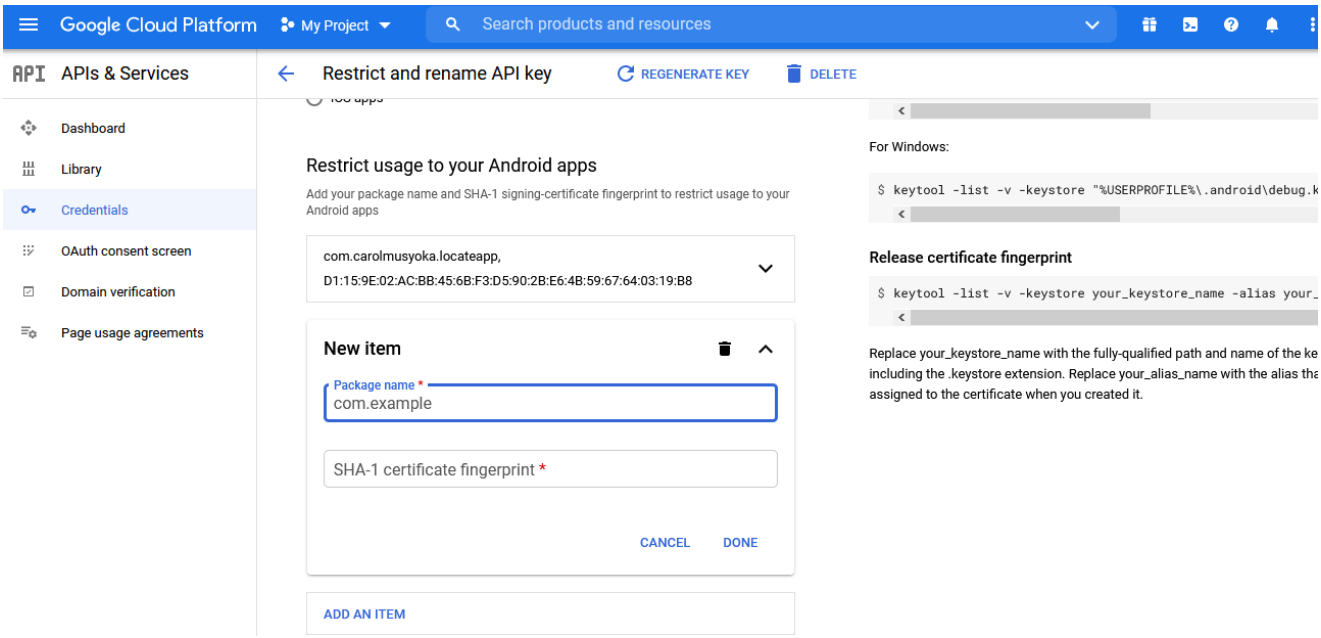
This second application allows you to retrieve the user's location from the database.

Step 1: Creating a new project

Follow the process discussed above to create a new project. Make sure you select the Google Maps template and name it appropriately.

Step 2: Add credentials to an existing key

Since we already have an API key, we can just include it in the console. Open your developer's console and click on the edit icon.



Step 3: Adding a button

Here is the **activity_maps.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
```

```
    xmlns:map="http://schemas.android.com/apk/res-auto"
    android:layout_height="match_parent"
    tools:context=".MapsActivity"
    xmlns:tools="http://schemas.android.com/tools">

<fragment
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    map:layout_constraintLeft_toLeftOf="parent"
    map:layout_constraintRight_toRightOf="parent"
    map:layout_constraintTop_toTopOf="parent"
    map:layout_constraintBottom_toBottomOf="parent" />
<Button
      android:layout_width="wrap_content"
        map:layout_constraintLeft_toLeftOf="parent"
        map:layout_constraintRight_toRightOf="parent"
        map:layout_constraintBottom_toBottomOf="parent"
        android:padding="20dp"
        android:id="@+id/btn_find_location"
        android:text="@string/find_user_s_location"
        android:layout_height="wrap_content" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 4: Adding permissions

By default, the GoogleMaps activity template adds the ACCESS_FINE_LOCATION permission in the AndroidManifest.xml file. Since we need the internet to read from the database, add the internet permission, as shown below:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Step 4: The model class

```
import com.google.firebase.database.IgnoreExtraProperties

@IgnoreExtraProperties

data class LocationInfo(

    var latitude: Double? = 0.0,

    var longitude: Double? = 0.0

)
```

Step 5: The MapsActivity

The following code:

```
  class MapsActivity : AppCompatActivity(), OnMapReadyCallback {

   private lateinit var map: GoogleMap

    private var database: FirebaseDatabase = FirebaseDatabase.getInstance()

    private var dbReference: DatabaseReference = database.getReference("test")

    private lateinit var find_location_btn: Button

       override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_maps)

 find_location_btn = findViewById(R.id.btn_find_location)
```

```
    // Obtain the SupportMapFragment and get notified when the map is ready to
be used.
        val mapFragment = supportFragmentManager
                .findFragmentById(R.id.map) as SupportMapFragment
        mapFragment.getMapAsync(this)
        // Get a reference from the database so that the app can read and write
operations
            dbReference = Firebase.database.reference
            dbReference.addValueEventListener(locListener)
    }
 val locListener = object : ValueEventListener {
        //     @SuppressLint("LongLogTag")
        override fun onDataChange(snapshot: DataSnapshot) {
            if(snapshot.exists()){
            //get the exact longitude and latitude from the database "test"
                val location =
snapshot.child("test").getValue(LocationInfo::class.java)
                val locationLat = location?.latitude
                val locationLong = location?.longitude
                //trigger reading of location from database using the button
                find_location_btn.setOnClickListener {
                  // check if the latitude and longitude is not null
                    if (locationLat != null && locationLong!= null) {
                    // create a LatLng object from location
                        val latLng = LatLng(locationLat, locationLong)
```

```
                    //create a marker at the read location and display it on the map
                    map.addMarker(MarkerOptions().position(latLng)
                            .title("The user is currently here"))
                            //specify how the map camera is updated
                    val update = CameraUpdateFactory.newLatLngZoom(latLng,
16.0f)
                    //update the camera with the CameraUpdate object
                    map.moveCamera(update)
                }
                else {
                    // if location is null , log an error message
                    Log.e(TAG, "user location cannot be found")
                }
            }
        }
    }
    // show this toast if there is an error while reading from the database
    override fun onCancelled(error: DatabaseError) {
        Toast.makeText(applicationContext, "Could not read from database",
Toast.LENGTH_LONG).show()
    } }
override fun onMapReady(googleMap: GoogleMap) {
    map = googleMap //initialize map when the map is ready }
  companion object {
```

```
    // TAG is passed into the Log.e methods used above to print information to
the Logcat window

    private const val TAG = "MapsActivity" // for debugging

}
```

SOURCE CODE

SignUpActivity.java

```
package com.example.covid_19alertapp.activities;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import android.Manifest;
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.example.covid_19alertapp.R;
import com.example.covid_19alertapp.extras.Constants;
import com.example.covid_19alertapp.extras.LogTags;
import com.example.covid_19alertapp.extras.Permissions;
```

```
import com.google.firebase.FirebaseException;
import com.google.firebase.auth.PhoneAuthCredential;
import com.google.firebase.auth.PhoneAuthProvider;

import java.util.concurrent.TimeUnit;
public class SignUpActivity extends AppCompatActivity {
Button btnContinue,btnHomeSignup,btnForwardSignup;
    EditText phoneNumber;
    TextView textViewTermsCond;
    public static String PHONE_NUMBER,verification;
    public static boolean ISRETURNEDFROMVERLAYOUT;
    public static SharedPreferences loginSp,userInfo;
    PhoneAuthProvider.OnVerificationStateChangedCallbacks mCallbacks;
  @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_up);
 // ask permissions
        promptPermissions();
        phoneNumber = findViewById(R.id.editText_phoneNumber);
        btnContinue = findViewById(R.id.btn_continue);
        textViewTermsCond = findViewById(R.id.TextViewTerm);
        btnHomeSignup = findViewById(R.id.home_button_signup_page);
 btnForwardSignup = findViewById(R.id.forward_button_signup_page);
loginSp=getSharedPreferences(Constants.USER_LOGIN_INFO_SHARED_PR
EFERENCES,MODE_PRIVATE);
userInfo=getSharedPreferences(Constants.USER_INFO_SHARED_PREFERE
NCES,MODE_PRIVATE);
if(loginSp.getBoolean(Constants.user_login_state_shared_preference
startActivity(new Intent(getApplicationContext(),
VerificationPageActivity.class));
            finish();
```

```
    }
mCallbacks=new PhoneAuthProvider.OnVerificationStateChangedCallbacks()
{
        @Override
 public void onVerificationCompleted(@NonNull PhoneAuthCredential
phoneAuthCredential) {

Toast.makeText(getApplicationContext(),"Successful",Toast.LENGTH_SHOR
T).show();
        }@Override
        public void onVerificationFailed(@NonNull FirebaseException e) {
Toast.makeText(getApplicationContext(),"Check Your Internet
Connection",Toast.LENGTH_SHORT).show();
            btnContinue.setEnabled(true); }
        @Override
        public void onCodeSent(@NonNull String s, @NonNull
PhoneAuthProvider.ForceResendingToken forceResendingToken) {
            super.onCodeSent(s, forceResendingToken);
            verification=sToast.makeText(getApplicationContext(),"Code Sent to
the Number",Toast.LENGTH_SHORT).show();
            startActivity(new Intent(getApplicationContext(),
VerificationPageActivity.class));

loginSp.edit().putBoolean(Constants.user_login_state_shared_preference,true).a
pply();
            btnContinue.setEnabled(true)
 finish();
        } };
if(ISRETURNEDFROMVERLAYOUT)
    {
        PHONE_NUMBER=PHONE_NUMBER.substring(0,4)+"
"+PHONE_NUMBER.substring(4);
        phoneNumber.setText(PHONE_NUMBER);
        ISRETURNEDFROMVERLAYOUT = false;
```

```
        btnHomeSignup.setVisibility(View.INVISIBLE);
        btnForwardSignup.setVisibility(View.VISIBLE);
        btnForwardSignup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(getApplicationContext(),
VerificationPageActivity.class));
loginSp.edit().putBoolean(Constants.user_login_state_shared_preference,true).a
pply();
 finish();
            }
        });
    }
 phoneNumber.clearFocus();
phoneNumber.setSelection(phoneNumber.getText().toString().length());
    phoneNumber.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int
after) {
        }
        //l
        int countB=phoneNumber.getText().toString().length(),countA=0;
        @SuppressLint("SetTextI18n")
  @Override
        public void onTextChanged(CharSequence s, int start, int before, int
count) {
if(phoneNumber.getText().toString().length()<5)
            {
                phoneNumber.setText("+880 ");

phoneNumber.setSelection(phoneNumber.getText().toString().length());
            }
```

```
            countA = phoneNumber.getText().toString().length();
           if(phoneNumber.getText().toString().length()==9 &&
countA>countB)
            {
               phoneNumber.setText(phoneNumber.getText().toString()+"-");

phoneNumber.setSelection(phoneNumber.getText().toString().length());
            }
            countB = countA;
            if(phoneNumber.getText().toString().length()==16)
            {
               hideSoftInput();
            }
         }
         @Override
         public void afterTextChanged(Editable s) { }
      });
 phoneNumber.setOnFocusChangeListener(new
View.OnFocusChangeListener() {
         @Override
         public void onFocusChange(View v, boolean hasFocus) {
            if(hasFocus)  phoneNumber.setCursorVisible(true);
            else phoneNumber.setCursorVisible(false);
         }
      });
 btnContinue.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            if(phoneNumber.getText().toString().length()==16) //Write a
function to check phone number validity
            {
               PHONE_NUMBER = phoneNumber.getText().toString();
```

```
                PHONE_NUMBER=PHONE_NUMBER.replaceAll("\\s+","");
                System.out.println(PHONE_NUMBER);

userInfo.edit().putString(Constants.user_phone_no_preference,PHONE_NUMB
ER).apply();
                sendSms(PHONE_NUMBER);
                btnContinue.setEnabled(false);


            }
            else
            {
                phoneNumber.setError("Invalid Number!");
            }
        }
    });


    textViewTermsCond.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //Write Terms and Condition Page Function

textViewTermsCond.setTextColor(getResources().getColor(R.color.colorInacti
ve));
        }
    });
btnHomeSignup.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            finish();
        }
    }); }
 public void hideSoftInput() {
```

```
        View view1 = this.getCurrentFocus();
        if(view1!= null){
            InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
            imm.hideSoftInputFromWindow(view1.getWindowToken(), 0);
        }
    }
public void sendSms(String phoneNo){
PhoneAuthProvider.getInstance().verifyPhoneNumber(
                phoneNo,        // Phone number to verify
                60,                // Timeout duration
                TimeUnit.SECONDS,   // Unit of timeout
                this,              // Activity (for callback binding)
                mCallbacks         // OnVerificationStateChangedCallbacks
        );
}
 /*
    permission needed at start of app
     */
 private Permissions permissions;
    private static final String[] permissionStrings = {
            Manifest.permission.ACCESS_FINE_LOCATION,
            Manifest.permission.ACCESS_BACKGROUND_LOCATION,
            Manifest.permission.ACCESS_WIFI_STATE,
            Manifest.permission.CALL_PHONE
    };
private void promptPermissions() {
permissions = new Permissions(this, permissionStrings,
Constants.PERMISSION_CODE);
 if(!permissions.checkPermissions())
            permissions.askPermissions();
 }
```

```
  @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        //resolve unresolved permission
switch (requestCode){
 case Constants.PERMISSION_CODE:
  try {
        this.permissions.resolvePermissions(permissions, grantResults);
              }catch (Exception e){
                 Log.d(LogTags.Permissions_TAG, "onRequestPermissionsResult:
"+e.getMessage());
              }
 break;
 } }}
```

UserInfoFormActivity.java

```
package com.example.covid_19alertapp.activities;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.graphics.drawable.Drawable;

import android.os.Bundle;

import android.os.Handler;

import android.util.Log;

import android.view.View;
```

```
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import com.example.covid_19alertapp.R;
import com.example.covid_19alertapp.extras.AddressReceiver;
import com.example.covid_19alertapp.models.UserInfoData;
import  com.example.covid_19alertapp.extras.Constants;
import com.example.covid_19alertapp.extras.LogTags;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
public class UserInfoFormActivity extends AppCompatActivity implements
AddressReceiver.AddressView {
    EditText dobText,userName;
    TextView workAddress,homeAddress;
    Button save_profile;
    UserInfoData userInfoData;
    FirebaseDatabase database;
    DatabaseReference userInfoRef;
    String uid= FirebaseAuth.getInstance().getCurrentUser().getUid();
    String path="UserInfo";
```

```
    public static SharedPreferences userInfo;

    private String homeLatLng = "", workLatLng = "", homeAddressVariable =
"", workAddressVariable = "";

   // address picker keys

    private static final int HOME_ADDRESS_PICKER = 829;

    private static final int WORK_ADDRESS_PICKER = 784;

 // address picker icon

    Drawable checkedIcon;

  // latLng to address fetcher

    AddressReceiver addressReceiver = new AddressReceiver(new Handler(),
this);

 @Override

 protected void onCreate(Bundle savedInstanceState) {

  super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_user_info_form);

       dobText= findViewById(R.id.dateOfBirth);

       userName = findViewById(R.id.userName);

       workAddress = findViewById(R.id.workAdress);

       homeAddress = findViewById(R.id.homeAddress);

       save_profile = findViewById(R.id.SaveProfButton);


userInfo=getSharedPreferences(Constants.USER_INFO_SHARED_PREFERE
NCES,MODE_PRIVATE);

checkedIcon=getApplicationContext().getResources().getDrawable(R.drawable.
ic_check_black_24dp);
```

```
        homeAddress.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
 // home address click
                Intent homeIntent = new Intent(UserInfoFormActivity.this, AddressPickerMapsActivity.class);
                 startActivityForResult(homeIntent,HOME_ADDRESS_PICKER); }
        });


        workAddress.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {


                // work address click
 Intent workIntent = new Intent(UserInfoFormActivity.this, AddressPickerMapsActivity.class);
                startActivityForResult(workIntent, WORK_ADDRESS_PICKER);
  }
        } save_profile.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(homeLatLng.equals("") || homeAddressVariable.equals("") || RequiredEditText(userName) || RequiredEditText(dobText))
                {
```

```
            if(homeLatLng.equals(""))
                homeAddress.setError("Required");
            else if(homeAddressVariable.equals("")) {
                Toast.makeText(UserInfoFormActivity.this,
                        "please wait as we fetch your address",
                        Toast.LENGTH_LONG)
                        .show();
            }

            return;
        }

        final String name,day,month,year,dateOfBirth,contactNumber;
        name=userName.getText().toString();
        dateOfBirth=dobText.getText().toString()
contactNumber=userInfo.getString(Constants.user_phone_no_preference,"Not
Defined
 if(workLatLng.equals("")){
            userInfoData=new
UserInfoData(name,dateOfBirth,homeLatLng,contactNumber,
homeAddressVariable)
  }
 else {
```

```
userInfoData = new UserInfoData(name, dateOfBirth, workLatLng, homeLatLng, contactNumber, homeAddressVariable, workAddressVariable);

userInfo.edit().putString(Constants.user_work_address_latlng_preference,workLatLng).apply();

userInfo.edit().putString(Constants.user_work_address_preference,workAddressVariable).apply();
        }
        //applying values to the info names Shared Preference
userInfo.edit().putString(Constants.username_preference,name).apply();
userInfo.edit().putString(Constants.username_preference,name).apply();

userInfo.edit().putString(Constants.user_dob_preference,dateOfBirth).apply();

userInfo.edit().putString(Constants.user_home_address_latlng_preference,homeLatLng).apply();
        userInfo.edit().putString(Constants.uid_preference,uid).apply();

//userInfo.edit().putString(Constants.user_phone_no_preference,PHONE_NUMBER).apply();

userInfo.edit().putBoolean(Constants.user_exists_preference,true).apply();

        // set the home address fetched using intent service
        userInfo.edit().putString(Constants.user_home_address_preference, homeAddressVariable).apply();
```

```
            database = FirebaseDatabase.getInstance();
            userInfoRef = database.getReference(path).child(uid);
            userInfoRef.setValue(userInfoData);



            startActivity(new                    Intent(getApplicationContext(),
MenuActivity.class));
            finish();
 }
  //Home Address field's onCLick function
   public void setHomeAddress(View v)
 }
   //Work ADDress field's onlick funcion
   public void setWorkAddress(View v){
  }
 @Override
   protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
       super.onActivityResult(requestCode, resultCode, data);
       /*
 receive latLong picked from map
 */
 switch (requestCode){
```

```
        case (HOME_ADDRESS_PICKER):
            if(resultCode == RESULT_OK){

                // set the home LatLng
                homeLatLng = data.getStringExtra("latitude-longitude");
                Log.d(LogTags.Map_TAG, "onActivityResult: home latLng fetched = "+homeLatLng);

                // start address fetch intent service
                String[] latLng = homeLatLng.split(",");
                addressReceiver.startAddressFetchService(
                        this,
                        Double.valueOf(latLng[0]),
                        Double.valueOf(latLng[1]),
                        0
                );

                //onSuccess
                homeAddress.setText(getText(R.string.address_fetching_text));
                homeAddress.setCompoundDrawables(null,null,checkedIcon,null);

            }
```

```
            break;


        case (WORK_ADDRESS_PICKER):
            if(resultCode == RESULT_OK){


                // set the work address
                workLatLng = data.getStringExtra("latitude-longitude");
                Log.d(LogTags.Map_TAG, "onActivityResult: work latLng fetched = "+workLatLng);


                // start address fetch intent service
                String[] latLng = workLatLng.split(",");
                addressReceiver.startAddressFetchService(
                        this,
                        Double.valueOf(latLng[0]),
                        Double.valueOf(latLng[1]),
                        1
                );


                //onSuccess
                workAddress.setCompoundDrawables(null,null,checkedIcon,null);
                workAddress.setText(getText(R.string.address_fetching_text));
```

```
}
break;
  }
 private boolean RequiredEditText(EditText e)
    {
        if(e.getText().toString().length()==0)
        {
            e.setError("Required");
            return true;
        }
return false;
    }
  @Override
    public void updateAddress(String address, int type) {
        /*
        address received callback
         */
 if(type == 0) {
            // home address
 homeAddressVariable = address;
homeAddress.setText(homeAddressVariable);
        }
           else if(type==1){
```

```
        // work address
        workAddressVariable = address;
        workAddress.setText(workAddressVariable);
}}}
```

VerificationPageActivity.java

```
package com.example.covid_19alertapp.activities;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Handler;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
```

```
import android.widget.Toast;
import androidx.appcompat.widget.Toolbar;

import com.example.covid_19alertapp.R;
import com.example.covid_19alertapp.extras.Constants;
import com.example.covid_19alertapp.models.UserInfoData;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.PhoneAuthCredential;
import com.google.firebase.auth.PhoneAuthProvider;
import com.google.firebase.auth.UserInfo;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import static com.example.covid_19alertapp.activities.SignUpActivity.verification;

public class VerificationPageActivity extends AppCompatActivity {
```

```
Toolbar toolbar;
    Button homeButton,confirmButton,editNumberButton;
    TextView textViewResendOTP;
    EditText digit1,digit2,digit3,digit4,digit5,digit6;
    String verificationCode,uid;
    FirebaseAuth auth;
    SharedPreferences sp,userInfoCheck,signUpSp;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_verification_page);
homeButton = findViewById(R.id.home_button_verification_page);
        toolbar = findViewById(R.id.verification_page_toolbar);
        digit1=findViewById(R.id.editTextDigit1);
        digit2=findViewById(R.id.editTextDigit2);
        digit3=findViewById(R.id.editTextDigit3);
        digit4=findViewById(R.id.editTextDigit4);
        digit5=findViewById(R.id.editTextDigit5);
        digit6=findViewById(R.id.editTextDigit6);
        auth=FirebaseAuth.getInstance();
       setSupportActionBar(toolbar);
       sp = getSharedPreferences("verify",MODE_PRIVATE);
        userInfoCheck=getSharedPreferences("info",MODE_PRIVATE);
```

```
        signUpSp                                                    =
getSharedPreferences(Constants.USER_LOGIN_INFO_SHARED_PREFEREN
CES,MODE_PRIVATE);

if(sp.getBoolean("logged",false)){

if(userInfoCheck.getBoolean(Constants.user_exists_preference,false)) {

              GoToMainActivity();

              finish();

            else

 checkIfUserInfoExist();

  }

homeButton.setOnClickListener(new View.OnClickListener() {

          @Override

          public void onClick(View v) {

              SignUpActivity.ISRETURNEDFROMVERLAYOUT = true;

startActivity(newIntent(getApplicationContext(),SignUpActivity.class)

finish();

          }

       });

((EditText)findViewById(R.id.editTextDigit1)).setCursorVisible(false);

       findViewById(R.id.editTextDigit1).setOnClickListener(new
View.OnClickListener() {

          @Override

          public void onClick(View v) {
```

```
            ((EditText)
findViewById(R.id.editTextDigit1)).setCursorVisible(true); }});


((EditText)findViewById(R.id.editTextDigit1)).addTextChangedListener(new
TextWatcher() {
        @Override
 {
   findViewById(R.id.editTextDigit2).clearFocus();
   findViewById(R.id.editTextDigit2).requestFocus();
              ((EditText)
findViewById(R.id.editTextDigit2)).setCursorVisible(true);
            }
        }
        @Override
        public void afterTextChanged(Editable s) { }
      });


((EditText)findViewById(R.id.editTextDigit2)).addTextChangedListener(new
TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int
after) { }
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int
count) {
```

```
if(((EditText)findViewById(R.id.editTextDigit2)).getText().toString().length()==1)
            {
               findViewById(R.id.editTextDigit3).clearFocus();
               findViewById(R.id.editTextDigit3).requestFocus();
               ((EditText) findViewById(R.id.editTextDigit3)).setCursorVisible(true);
            }
         }
         @Override
         public void afterTextChanged(Editable s) { }
      });

((EditText)findViewById(R.id.editTextDigit3)).addTextChangedListener(new TextWatcher() {
         @Override
         public void beforeTextChanged(CharSequence s, int start, int count, int after) { }
         @Override
         public void onTextChanged(CharSequence s, int start, int before, int count) {

if(((EditText)findViewById(R.id.editTextDigit3)).getText().toString().length()==1)
            {
```

```
                findViewById(R.id.editTextDigit4).clearFocus();
                findViewById(R.id.editTextDigit4).requestFocus();
                ((EditText)
findViewById(R.id.editTextDigit4)).setCursorVisible(true);
            }
        }
        @Override
        public void afterTextChanged(Editable s) { }
    });

((EditText)findViewById(R.id.editTextDigit4)).addTextChangedListener(new
TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int
after) { }
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int
count) {

if(((EditText)findViewById(R.id.editTextDigit4)).getText().toString().length()=
=1)
            {
                findViewById(R.id.editTextDigit5).clearFocus();
                findViewById(R.id.editTextDigit5).requestFocus();
                ((EditText)
findViewById(R.id.editTextDigit5)).setCursorVisible(true);
```

```
            }
        }
        @Override
        public void afterTextChanged(Editable s) { }
    });

((EditText)findViewById(R.id.editTextDigit5)).addTextChangedListener(new
TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int
after) { }
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int
count) {

if(((EditText)findViewById(R.id.editTextDigit5)).getText().toString().length()=
=1)
            {
                findViewById(R.id.editTextDigit6).clearFocus();
                findViewById(R.id.editTextDigit6).requestFocus();
                ((EditText)
findViewById(R.id.editTextDigit6)).setCursorVisible(true);
            }
        }
        @Override
```

```
        public void afterTextChanged(Editable s) { }
    });

((EditText)findViewById(R.id.editTextDigit6)).addTextChangedListener(new
TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int
after) { }
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int
count) {

if(((EditText)findViewById(R.id.editTextDigit6)).getText().toString().length()=
=1)
            {
                ((EditText)
findViewById(R.id.editTextDigit6)).setCursorVisible(false);
                findViewById(R.id.btn_continue).clearFocus();
                findViewById(R.id.btn_continue).requestFocus();
                hideSoftInput();
            }
        }
        @Override
        public void afterTextChanged(Editable s) { }
    });
```

```
textViewResendOTP = findViewById(R.id.TextViewResendOTP);
        textViewResendOTP.setOnClickListener(new View.OnClickListener() {
@Overr
public void onClick(View v) {
                Toast.makeText(getApplicationContext(),"RESENDING OTP",Toast.LENGTH_SHORT).show();
                textViewResendOTP.setEnabled(false);
textViewResendOTP.setTextColor(getResources().getColor(R.color.colorInactive));
                ToggleResendTextView(textViewResendOTP);
                //Write ResendOTP Function Here
}
        });
       ToggleResendTextView(textViewResendOTP);
        confirmButton = findViewById(R.id.btn_continue);
        editNumberButton = findViewById(R.id.btn_change_number);
        confirmButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Write OTP Submission Function Here

verificationCode=digit1.getText().toString().trim()+""+digit2.getText().toString().trim()+""+digit3.getText().toString().trim()+""+digit4.getText().toString().trim()+""+digit5.getText().toString().trim()+""+digit6.getText().toString().trim();
```

```
            System.out.println(verificationCode+"                    sdf"+
digit1.getText().toString());
            verify(verificationCode);




        }
    });


    editNumberButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            startActivity(new
Intent(getApplicationContext(),SignUpActivity.class));
            SignUpActivity.ISRETURNEDFROMVERLAYOUT = true;

signUpSp.edit().putBoolean(Constants.user_login_state_shared_preference,fals
e).apply();
            finish();


        }
    });
```

```
    }


    //Methods
    public void hideSoftInput() {
        View view1 = this.getCurrentFocus();
        if(view1!= null){
            InputMethodManager imm = (InputMethodManager) getSystemService(Context.INPUT_METHOD_SERVICE);
            imm.hideSoftInputFromWindow(view1.getWindowToken(), 0);
        }
    }
    public void ToggleResendTextView(final TextView textView)
    {
        final Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                textView.setEnabled(true);

textView.setTextColor(getResources().getColor(R.color.colorActive));
            }
        }, 20000);
    }
```

```
public void verify(String verificationCode){
    System.out.println(verification+" verify");
    verfyPhoneNumber(verification,verificationCode);
}

private void verfyPhoneNumber(String verification, String enterededCodeString) {
    System.out.println(verification+" credential "+enterededCodeString);
    PhoneAuthCredential phoneAuthCredential= PhoneAuthProvider.getCredential(verification,enterededCodeString);
    signInWithPhoneAuthCredential(phoneAuthCredential);
}

private void signInWithPhoneAuthCredential(PhoneAuthCredential credential) {
    auth.signInWithCredential(credential)
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    // Sign in success, update UI with the signed-in user's information
                    //Log.d(TAG, "signInWithCredential:success");
```

```
                    //System.out.println("Successful");
                    FirebaseUser user = task.getResult().getUser();
                    uid= FirebaseAuth.getInstance().getCurrentUser().getUid();

if(userInfoCheck.getBoolean(Constants.user_exists_preference,false)) {
                       GoToMainActivity();
                       finish();
                    }
                    else
                       checkIfUserInfoExist();

                    sp.edit().putBoolean("logged",true).apply();
                    Toast.makeText(getApplicationContext(),"User Signed In Successfully",Toast.LENGTH_SHORT).show();

                } else {
                    //System.out.println(task.getException()+" task exception");
                    Toast.makeText(getApplicationContext(),"Please use the valid code",Toast.LENGTH_SHORT).show();
                    // Sign in failed, display a message and update the UI

                }
            }
```

```
            });
    }

    public void checkIfUserInfoExist(){

        FirebaseDatabase database = FirebaseDatabase.getInstance();

        uid=FirebaseAuth.getInstance().getUid();

        DatabaseReference ref = database.getReference().child("UserInfo").child(uid);

        ValueEventListener valueEventListener = new ValueEventListener() {

            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                if(dataSnapshot.exists()){

                    UserInfoData user = dataSnapshot.getValue(UserInfoData.class);

                    userInfoCheck.edit().putString(Constants.username_preference, user.getName()).apply();
                    userInfoCheck.edit().putString(Constants.user_dob_preference, user.getDob()).apply();
```

```
userInfoCheck.edit().putString(Constants.user_home_address_preference,
user.getHomeAddress()).apply();


userInfoCheck.edit().putString(Constants.user_home_address_latlng_preference
, user.getHomeLatLng()).apply();

userInfoCheck.edit().putString(Constants.uid_preference,uid).apply();


userInfoCheck.edit().putString(Constants.user_phone_no_preference,
user.getContactNumber()).apply();

userInfoCheck.edit().putBoolean(Constants.user_exists_preference,true).apply()
;

if(String.valueOf(dataSnapshot.child(Constants.userInfo_node_workAddress).g
etValue())!=null) {


userInfoCheck.edit().putString(Constants.user_work_address_preference,
user.getWorkAddress()).apply();


userInfoCheck.edit().putString(Constants.user_work_address_latlng_preference
, user.getWorkLatLng()).apply();
                }


                GoToMainActivity();

            }
```

```
            else {


                GotoUserInfoFormActivity();
            }finish();
        }


        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
  }
  };
      ref.addListenerForSingleValueEvent(valueEventListener);
 }
public void GoToMainActivity(){


   startActivity(newIntent(getApplicationContext(), MenuActivity.class));
   }
   public void GotoUserInfoFormActivity(){


      startActivity(nIntent(getApplicationContext(),
UserInfoFormActivity.class));
   }
}
```

MenuActivity.java

```
package com.example.covid_19alertapp.activities;

import androidx.annotation.Nullable;

import androidx.appcompat.app.AlertDialog;

import androidx.appcompat.app.AppCompatActivity;

import androidx.lifecycle.Observer;

import androidx.work.Constraints;

import androidx.work.PeriodicWorkRequest;

import androidx.work.WorkInfo;

import androidx.work.WorkManager;


import android.content.DialogInterface;

import android.content.Intent;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.Button;


import com.example.covid_19alertapp.R;

import com.example.covid_19alertapp.extras.Constants;

import com.example.covid_19alertapp.extras.LogTags;

import com.example.covid_19alertapp.services.BackgroundWorker;
```

```
import
com.example.covid_19alertapp.sharedPreferences.MiscSharedPreferences;


import java.util.List;
import java.util.concurrent.TimeUnit;



public class MenuActivity extends AppCompatActivity {
    /*
    starter activity to test and get the permissions + all time running start worker
    overwrite or edit this later, keeping the permission codes
     */

    Button home_btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        home_btn = findViewById(R.id.home_button_menu);
        home_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
            finish();
        }
    });
    // start background worker for always
    startWorker()
  }
private void startWorker() {
if(!MiscSharedPreferences.getBgWorkerStatus(this)){
Constraints constraints = new Constraints.Builder()
            .setRequiresBatteryNotLow(true)
            .setRequiresCharging(false)
            .build();
 PeriodicWorkRequest promptNotificationWork =
            NewPeriodicWorkRequest.Builder(BackgroundWorker.class,  30,
TimeUnit.MINUTES)
                .setConstraints(constraints)
                .addTag(Constants.background_WorkerTag)
                .build();
WorkManager.getInstance(getApplicationContext()).getWorkInfoByIdLiveData
(promptNotificationWork.getId())
            .observe(this, new Observer<WorkInfo>() {
                @Override
 public void onChanged(@Nullable WorkInfo workInfo) {
```

```
                    if (workInfo != null && workInfo.getState() ==
WorkInfo.State.ENQUEUED) {
                        Log.d(LogTags.Worker_TAG, "onChanged: worker is
enqueued");
 // set shared preference true
MiscSharedPreferences.setBgWorkerStatus(MenuActivity.this, true);
                    }
if (workInfo != null && workInfo.getState() == WorkInfo.State.CANCELLED)
{
                        Log.d(LogTags.Worker_TAG, "onChanged: worker was
stopped. why?");

                        // set shared preference false

MiscSharedPreferences.setBgWorkerStatus(MenuActivity.this, false);
                    }
                }
            });
 WorkManager.getInstance(getApplicationContext())
                .enqueue(promptNotificationWork);
  } }
public void uploadClick(View view) {
 if(!MiscSharedPreferences.getUploadStatus(this)) {
  Intent intent = new Intent(this, UploadLocationsActivity.class);
        startActivity(intent);
```

```
 }
     else{
 // show dialog and prevent
AlertDialog.Builder builder = new AlertDialog.Builder(this);
 builder.setMessage(getText(R.string.cant_upload_twice_message))
                .setCancelable(false)

.setPositiveButton(getText(R.string.permissions_dialogbox_positive), new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.dismiss();
                    }
                })
                .setNegativeButton("Override", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.dismiss();

                        // TODO: remove this
                        Intent intent = new Intent(MenuActivity.this,
UploadLocationsActivity.class);
                        startActivity(intent);
```

```
                    }
                });

            AlertDialog alertDialog = builder.create();
            alertDialog.show();

        }
    }

    public void startNewsFeed(View view)
    {
        startActivity(new
Intent(getApplicationContext(),NewsFeedActivity.class));
    }

    public void openSettingsClick(View view) {

        Intent intent = new Intent(this, TrackerSettingsActivity.class);
        startActivity(intent);

    }

    public void showMatchedLocationsClick(View view) {
```

```
 Intent intent = new Intent(getApplicationContext(), ShowMatchedLocationsActivity.class);
      startActivity(intent);  }
public void startMyLocationsMap(View view) {
startActivity( new Intent(this, MyLocationsMapsActivity.class) );
 }
}
```

UploadLocationsActivity.java

```
package com.example.covid_19alertapp.activities;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.Observer;
import android.content.DialogInterface;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;
```

```
import com.example.covid_19alertapp.R;

import com.example.covid_19alertapp.extras.Constants;

import com.example.covid_19alertapp.extras.LogTags;

import com.example.covid_19alertapp.models.InfectedLocations;

import com.example.covid_19alertapp.roomdatabase.LocalDBContainer;

import com.example.covid_19alertapp.roomdatabase.VisitedLocations;

import  com.example.covid_19alertapp.roomdatabase.VisitedLocationsDao;

import
com.example.covid_19alertapp.roomdatabase.VisitedLocationsDatabase;

import
com.example.covid_19alertapp.sharedPreferences.MiscSharedPreferences;

import
com.example.covid_19alertapp.sharedPreferences.UserInfoSharedPreferences;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseException;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;


import java.util.ArrayList;

import  java.util.Calendar;

import java.util.List;
```

```
public class UploadLocationsActivity extends AppCompatActivity {
/*
upload locations from local db to firebase
implement verification by medical report photo here
 */
 // firebase
    //private FirebaseDatabase firebaseDatabase;
    private DatabaseReference firbaseReference;

    // local db
    private VisitedLocationsDatabase roomDatabase;
    private VisitedLocationsDao visitedLocationsDao;

    // retrieved data from local db
    private List<VisitedLocations> retrievedDatas = new ArrayList<>();

    // retrieve and upload progress level
    private int dataSize, dataCount = 0;
    private double currProgress = 0;

    // models to store in firebase
```

```
    private MutableLiveData<InfectedLocations> currentInfectedLocation = new
MutableLiveData<>();

    final Observer<InfectedLocations> newEntryObserver = new
Observer<InfectedLocations>() {

        @Override

        public void onChanged(final InfectedLocations infectedLocations) {


            if(!infectedLocations.allFieldsSet()) {

                // exit if all values not set

                Log.d(LogTags.Upload_TAG, "onChanged: all fields not set");

                return;

            }


            // upload to firebase

            insertToFirebase("infectedLocations", infectedLocations.getKey(),
infectedLocations.getDateTime(), infectedLocations.getCount());


            // blacklist user


            // get user uid

            String uid =
UserInfoSharedPreferences.getUid(UploadLocationsActivity.this);


            insertToFirebase("blackList/"+uid+"/visitedLocations",
```

```
                infectedLocations.getKey(), infectedLocations.getDateTime(),
infectedLocations.getCount());


        }
    };

    // UI stuff
    ProgressBar uploadProgressBar;
    TextView uploadProgressText;
    Button uploadButton, home_btn;

    // back press during uploading
    boolean uploading = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_upload_locations);
        home_btn = findViewById(R.id.home_button_upload_locations);
        home_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
```

```
        }
    });

    setUpUI();

    // set firebase database offline capability, set firebase reference
    if(firbaseReference == null) {
        FirebaseDatabase database = FirebaseDatabase.getInstance();
        try {
            database.setPersistenceEnabled(true);
        }catch (DatabaseException e){
            Log.d(LogTags.Upload_TAG, "onCreate: setPersistent issue. need to fix this");
        }
        firbaseReference  = database.getReference();
    }

    // set local db configs
    roomDatabase = VisitedLocationsDatabase.getDatabase(getApplicationContext());
    visitedLocationsDao = roomDatabase.visitedLocationsDao();

    // set InfectedLocation Live Data observer
    currentInfectedLocation.observe(this, newEntryObserver);
```

```
    }


    @Override
    public void onBackPressed() {


        if(uploading) {


            // show dialog
            Log.d(LogTags.Upload_TAG, "onBackPressed: back pressed during uploading");


            AlertDialog.Builder builder = new AlertDialog.Builder(this);


            builder.setMessage(getText(R.string.backPressed_during_upload))
                    .setCancelable(false)

.setPositiveButton(getText(R.string.backPressed_during_upload_positive), new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
```

```
                    dialog.dismiss();
                    Log.d(LogTags.Upload_TAG, "onClick: uploading resumes");
                }
            });

        AlertDialog alertDialog = builder.create();
        alertDialog.show();

    }
    else
        super.onBackPressed();

}

private void setUpUI() {

    uploadProgressBar = findViewById(R.id.uploadProg
```
ressBar);
```
    uploadProgressText = findViewById(R.id.uploadProgressText);
    uploadButton = findViewById(R.id.upload_btn);

}

private void uploadAndDeleteLocal() {
```

```
/*
retrive from local database,
upload to firebase,
delete from local databse
 */

// save the uploading state
uploading = true;
uploadProgressText.setVisibility(View.VISIBLE);
uploadProgressBar.setVisibility(View.VISIBLE);

roomDatabase.databaseWriteExecutor.execute(new Runnable() {
    @Override
    public void run() {

        // fetch all from localDB
        retrievedDatas = visitedLocationsDao.fetchAll();
        Log.d(LogTags.Upload_TAG, "onCreate: local database retrieved");

        // retrieval from localDB done (50%)
        currProgress = 50;
        dataSize = retrievedDatas.size();
```

```
            if(dataSize==0) {
                // notify on UI thread no data found locally
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        Toast.makeText(UploadLocationsActivity.this, "No locations recorded, only home address uploaded", Toast.LENGTH_LONG)
                                .show();
                        uploadProgressText.setVisibility(View.GONE);
                        uploadProgressBar.setVisibility(View.GONE);
                    }
                });

                uploading = false;

                return;
            }


            for(VisitedLocations roomEntry: retrievedDatas){

                // splitData[0] = lat,lon
```

```
                // splitData[1] = dateTime
                String[] splitData = roomEntry.splitPrimaryKey();

                Log.d(LogTags.Upload_TAG, "run: current retrieved data = "
                        +splitData[0]+", "+roomEntry.getCount()+", "+splitData[1]);

                // set the LiveData object
                currentInfectedLocation.postValue(new
InfectedLocations(splitData[0], roomEntry.getCount(), splitData[1]));

                // delete current entry from local database
                visitedLocationsDao.deleteLocation(roomEntry);
                Log.d(LogTags.Upload_TAG, "onCreate: deleting room entry = "
                        +roomEntry.getConatainerDateTimeComposite());

                // keep track of upload progress (50%-100%)
                currProgress += (double) 50/dataSize;
                uploadProgressBar.setProgress((int) currProgress);

                dataCount++;
                if(dataCount==dataSize){

                   runOnUiThread(new Runnable() {
```

```
                @Override
                public void run() {
                    // remove progressbar
uploadProgressText.setText(getText(R.string.uploadFinished_progressbar_text)
);
                        uploadProgressBar.setVisibility(View.GONE);
                    }
                });

                // uploading done
                uploading = false;

                // set upload status shared preference true
MiscSharedPreferences.setUploadStatus(UploadLocationsActivity.this, true);

            }

            // sleep, give time to upload properly?
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
```

```
                Log.d(LogTags.Upload_TAG, "run: thread just had coffee and isn't tired rn");
                e.printStackTrace();
            }
        }


    }
});


}

private void uploadHomeLocation(){

    List<String> entries;

    String homeLatLng = UserInfoSharedPreferences.getHomeLatLng(this);
    if(homeLatLng.equals("")){
        Log.d(LogTags.Upload_TAG, "uploadHomeLocation: why the hell is home null");
        return;
    }

    String[] latLng = homeLatLng.split(",");
```

```
entries = LocalDBContainer.calculateContainer(Double.parseDouble(latLng[0]), Double.parseDouble(latLng[1]), "Bangladesh");


// get current time
Calendar cal = Calendar.getInstance();
//TODO: add year
final String dateTime = (cal.get(Calendar.MONTH)+1) +"-" // Calender.MONTH is 0 based -_- why tf?
        + cal.get(Calendar.DATE) +"-"
        + cal.get(Calendar.HOUR_OF_DAY);


for (String entry: entries) {

    // need '@' instead of '.'
    entry = entry.replaceAll("\\.","@");

    // upload home address
    insertToFirebase("infectedHomes", entry, dateTime, 1);

    // blacklist user

    // get user uid
    String uid = UserInfoSharedPreferences.getUid(this);
```

```
            insertToFirebase("blackList/"+uid+"/home", entry, dateTime, 1);


        }


    }

    private void insertToFirebase(final String node, String latLon, String dateTime, final long count){


        final DatabaseReference currentReference = firbaseReference.child(node).child(latLon).child(dateTime);


        currentReference.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {



                if(dataSnapshot.child("unverifiedCount").getValue()!=null){
                    // data already exists
                    Log.d(LogTags.Upload_TAG, "onDataChange: location already exists at "+node);
```

```
                long existingCount = (long)dataSnapshot.child("unverifiedCount").getValue();

                currentReference.child("unverifiedCount").setValue(count + existingCount);

            }

            else{
                // no such data exists
                Log.d(LogTags.Upload_TAG, "onDataChange: new location at "+node);

                currentReference.child("unverifiedCount").setValue(count);
            }

            if(dataSnapshot.child("verifiedCount").getValue()==null)
                currentReference.child("verifiedCount").setValue(0);

        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
```

```
Log.d(LogTags.Upload_TAG, "onCancelled: firebase e somossa ki korbo? "+databaseError.getMessage() +", "+databaseError.getDetails());

 Toast.makeText(getApplicationContext(),

   getApplicationContext().getString(R.string.no_internet_toast),

                   Toast.LENGTH_LONG)

                   .show();


          }

       });


    }



    public void uploadClicked(View view) {

       /*

       upload button click

        */


       // show dialog before uploading


       AlertDialog.Builder builder = new AlertDialog.Builder(this);


       builder.setTitle(getText(R.string.upload_confirmation_title))
```

```
            .setMessage(getText(R.string.upload_confirmation_message))
            .setCancelable(false)
            .setPositiveButton(getText(R.string.upload_confirmation_positive),
new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.dismiss();

                    Log.d(LogTags.Upload_TAG, "onClick: uploading starts");

                    // upload home location
                    uploadHomeLocation();

                    // start uploading process
                    uploadButton.setEnabled(false);
                    uploadAndDeleteLocal();


                }
            })
            .setNegativeButton(getText(R.string.upload_confirmation_negative),
new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
```

```
                dialog.dismiss();

                // close the activity
                UploadLocationsActivity.this.finish();
                Log.d(LogTags.Upload_TAG, "onClick: not gonna upload");
            }
        });

    AlertDialog alertDialog = builder.create();
    alertDialog.show();

  }
}
```

ShowMatchedLocationsActivity.java

```
package com.example.covid_19alertapp.activities;

import androidx.annotation.NonNull;
import  androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;
import android.os.Handler;
```

```
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.example.covid_19alertapp.R;
import com.example.covid_19alertapp.adapters.LocationListAdapter;
import com.example.covid_19alertapp.extras.AddressReceiver;
import com.example.covid_19alertapp.extras.Constants;
import com.example.covid_19alertapp.extras.Internet;
import com.example.covid_19alertapp.extras.LogTags;
import com.example.covid_19alertapp.extras.Notifications;
import com.example.covid_19alertapp.models.MatchedLocation;
import com.example.covid_19alertapp.roomdatabase.LocalDBContainer;
import com.example.covid_19alertapp.roomdatabase.VisitedLocations;
import com.example.covid_19alertapp.roomdatabase.VisitedLocationsDao;
import
com.example.covid_19alertapp.roomdatabase.VisitedLocationsDatabase;
import
com.example.covid_19alertapp.sharedPreferences.UserInfoSharedPreferences;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
```

```
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

public class ShowMatchedLocationsActivity extends AppCompatActivity implements AddressReceiver.AddressView {

    // matched locations model (for recycler-view)
    ArrayList<MatchedLocation> matchedLocations = new ArrayList<>();
    int matchedLocationPosition = 0, locationQueryCount = 0;

    // matched home locations model (for another(?) recycler-view)
    ArrayList<MatchedLocation> matchedHomeLocations = new ArrayList<>();
    int homeQueryCount = 0;

    // firebase
    private DatabaseReference firebaseReference;

    // local db
    private VisitedLocationsDatabase roomDatabase;
```

```
    private VisitedLocationsDao visitedLocationsDao;

    // retrieved data from local db
    private List<VisitedLocations> retrievedDatas = new ArrayList<>();
    private int dataSize;

    // Address Fetch
    AddressReceiver addressReceiver = new AddressReceiver(new Handler(),
this);

    // UI stuff
    private ProgressBar progressBar;
    private TextView progressBarText;
    private Button retryButton;
    private RecyclerView locationRecyclerView, homeLocationRecyclerView;
    private LocationListAdapter locationListAdapter, homeLocationListAdapter;
    private boolean internetAvailable = true;

    // flags
    private boolean localDbEmptyFlag = false;
    private boolean homeLocationsFetchFinishedFlag = false;
    private boolean locationsFetchFinishedFlag = false;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_show_matched_locations);


        setUI();


        Notifications.removeNotification(Constants.DangerNotification_ID, this);


        // set local db configs
        roomDatabase                                                    =
VisitedLocationsDatabase.getDatabase(getApplicationContext());
        visitedLocationsDao = roomDatabase.visitedLocationsDao();


        // firebase
        firebaseReference = FirebaseDatabase.getInstance().getReference();


        findHomeMatchedLocations();
        findMatchedLocations();


    }
```

```
    private void setUI() {

        progressBar = findViewById(R.id.progressBar);
        progressBarText = findViewById(R.id.progressText);
        retryButton = findViewById(R.id.retry_btn);

        homeLocationRecyclerView = findViewById(R.id.homeRecyclerView);
        homeLocationRecyclerView.setLayoutManager(new
LinearLayoutManager(this));

        locationRecyclerView = findViewById(R.id.locationRecyclerView);
        locationRecyclerView.setLayoutManager(new
LinearLayoutManager(this));

    }

    private void findHomeMatchedLocations() {

        homeLocationsFetchFinishedFlag = false;
        matchedHomeLocations.clear();
        homeQueryCount = 0;
        homeLocationListAdapter = new LocationListAdapter(this,
matchedHomeLocations);
        homeLocationRecyclerView.setAdapter(homeLocationListAdapter);
```

```
List<String> queryKeys;

final String homeLatLng = UserInfoSharedPreferences.getHomeLatLng(this);
if(homeLatLng.equals("")){
    Log.d(LogTags.Worker_TAG, "queryHomeAddress: why the hell is home null");
    return;
}

final String[] latLng = homeLatLng.split(",");

queryKeys = LocalDBContainer.calculateContainer(Double.parseDouble(latLng[0]), Double.parseDouble(latLng[1]), "Bangladesh");

final int querySize = queryKeys.size();

for (String query: queryKeys) {

    if(!Internet.isInternetAvailable(getApplicationContext())){
        runOnUiThread(new Runnable() {
```

```
            @Override
            public void run() {

                internetDisconncetedUI();

            }
        });
        return;
    }

    // need '@' instead of '.'
    query = query.replaceAll("\\.","@");

    firebaseReference.child("infectedHomes").child(query)
            .addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                    if(dataSnapshot.getValue()!=null){

                        long verifiedCount = 0, unverifiedCount = 0;
                        for (DataSnapshot snapshot: dataSnapshot.getChildren()) {
```

```
verifiedCount+=(long)snapshot.child("verifiedCount").getValue();
                        unverifiedCount+=(long)
snapshot.child("unverifiedCount").getValue();
                    }

                    MatchedLocation homeLocation = new MatchedLocation(
                        Double.parseDouble(latLng[0]),
                        Double.parseDouble(latLng[1]),
                        "NEAR YOUR HOME!",
                        verifiedCount,
                        unverifiedCount
                    );

                    if(matchedHomeLocations.isEmpty()) {
                       // only find one match for home

                       matchedHomeLocations.add(homeLocation);

homeLocationListAdapter.notifyItemInserted(matchedHomeLocations.size()    -
1);

                       homeLocationsFetchFinishedFlag = true;
```

```
                    if(locationsFetchFinishedFlag)
                        dataFetchFinishedUI();
                    else if(localDbEmptyFlag)
                        localDbEmptyUI();
                }
                Log.d(LogTags.MatchFound_TAG, "onDataChange: home location matched: "+homeLocation.toString());

            }

            homeQueryCount++;
            if(homeQueryCount>=querySize){
                homeLocationsFetchFinishedFlag = true;

                if(locationsFetchFinishedFlag)
                    dataFetchFinishedUI();
                else if(localDbEmptyFlag)
                    localDbEmptyUI();
            }

        }
```

```
                @Override
                public void onCancelled(@NonNull DatabaseError databaseError) {

                    internetDisconncetedUI();

                    Log.d(LogTags.MatchFound_TAG, "onCancelled: home location query failed "+databaseError.getMessage());

                }
            });

        }

    }

    private void findMatchedLocations() {

        localDbEmptyFlag = false;
        locationsFetchFinishedFlag = false;
        matchedLocationPosition = 0;
        locationQueryCount = 0;

        if(internetAvailable) {
```

```
    retryButton.setVisibility(View.GONE);
    retryButton.setEnabled(false);
}
matchedLocations.clear();
locationListAdapter = new LocationListAdapter(this, matchedLocations);
locationRecyclerView.setAdapter(locationListAdapter);



roomDatabase.databaseWriteExecutor.execute(new Runnable() {
    @Override
    public void run() {



        // fetch from local db and query firebase
        retrievedDatas = visitedLocationsDao.fetchAll();


        dataSize = retrievedDatas.size();


        if(dataSize==0){
            // local database empty


            localDbEmptyFlag = true;
```

```
    if(homeLocationsFetchFinishedFlag) {

        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                localDbEmptyUI();
            }
        });
    }

    return;

}



for (VisitedLocations currentEntry: retrievedDatas)
{
    // format = "latLon_dateTime"
    String[] splitter = currentEntry.splitPrimaryKey();

    // firebase query values
    final String key = currentEntry.getATencodedlatlon();
    final String dateTime = splitter[1];
```

```
            Log.d(LogTags.MatchFound_TAG, "run: query key = "+key +"
date time = "+dateTime);


            if(!Internet.isInternetAvailable(getApplicationContext())){


                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {


                        internetDisconncetedUI();


                    }
                });


                return;
            }


            // query in firebase
            firebaseReference                                                =
FirebaseDatabase.getInstance().getReference().child("infectedLocations").child(
key).child(dateTime);
            firebaseReference.addListenerForSingleValueEvent(new
ValueEventListener() {
```

```
@Override
public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
    if(dataSnapshot.getValue()!=null){
        // INFECTED LOCATION MATCH FOUND!

        String latLon = key;
        long verifiedCount = (long) dataSnapshot.child("verifiedCount").getValue();
        long unverifiedCount = (long) dataSnapshot.child("unverifiedCount").getValue();

        MatchedLocation matchedLocation = new MatchedLocation(latLon, dateTime, verifiedCount, unverifiedCount);
        matchedLocations.add(matchedLocation);

locationListAdapter.notifyItemInserted(matchedLocationPosition);

        // start address fetch service
        addressReceiver.startAddressFetchService(
            ShowMatchedLocationsActivity.this,
            matchedLocation.getBlLatitude(),
            matchedLocation.getBlLongitude(),
            matchedLocationPosition
```

```
    );

    matchedLocationPosition++;

}

locationQueryCount++;
if(locationQueryCount>=dataSize){

    if(matchedLocations.isEmpty()){
        // no locations match

        locationsFetchFinishedFlag = true;

        if(matchedHomeLocations.isEmpty()) {
            // no home locations match either
            // show no match found

            runOnUiThread(new Runnable() {
                @Override
                public void run() {

                    noMatchFoundUI();
```

```
                }
            });
        }

        else {
            // no location match
            // but home location matched show finish UI

            runOnUiThread(new Runnable() {
                @Override
                public void run() {

                    dataFetchFinishedUI();

                }
            });

        }

    }

}
```

```
                }

                @Override
                public void onCancelled(@NonNull DatabaseError
databaseError) {

                    // internet connection lost

                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            internetDisconncetedUI();
                        }
                    }); }
                }); } }
        });}
 private void internetDisconncetedUI()
 internetAvailable = false;
 progressBar.setVisibility(View.INVISIBLE);
        //linearLayout.setVisibility(View.INVISIBLE);
     progressBarText.setText(getText(R.string.internet_disconnected_text));
        progressBarText.setVisibility(View.VISIBLE);
```

```
      retryButton.setEnabled(true);
       retryButton.setVisibility(View.VISIBLE);
       Log.d("removethis", "internetDisconncetedUI: visible");

        Toast.makeText(this,getText(R.string.no_internet_toast),
Toast.LENGTH_LONG)
                .show();

    }

    private void dataFetchFinishedUI(){

        retryButton.setEnabled(false);
        progressBarText.setVisibility(View.GONE);
        progressBar.setVisibility(View.GONE);
        if(internetAvailable) {
            retryButton.setVisibility(View.GONE);
            retryButton.setEnabled(false);
        }
        Toast.makeText(this,              getText(R.string.finished_progressbar_text),
Toast.LENGTH_LONG)
                .show();

    }
```

```
private void noMatchFoundUI(){

    progressBar.setVisibility(View.INVISIBLE);
    if(internetAvailable) {
        retryButton.setVisibility(View.GONE);
        retryButton.setEnabled(false);
    }
    progressBarText.setVisibility(View.VISIBLE);
    progressBarText.setText(getText(R.string.no_match_found_text));
}

private void localDbEmptyUI(){

    progressBar.setVisibility(View.INVISIBLE);
    //linearLayout.setVisibility(View.INVISIBLE);
    if(internetAvailable) {
        retryButton.setVisibility(View.GONE);
        retryButton.setEnabled(false);
    }
    progressBarText.setVisibility(View.VISIBLE);
    progressBarText.setText(getText(R.string.local_db_empty_text));
```

```
}

public void retryClicked(View view) {

    internetAvailable = true;

    progressBar.setVisibility(View.VISIBLE);
    progressBarText.setVisibility(View.VISIBLE);
    progressBarText.setText(getText(R.string.loading_progressbar_text));

    findHomeMatchedLocations();
    findMatchedLocations();

}

private int updateCount = 0;
@Override
public void updateAddress(String address, int listPosition) {
    /*
    address received here
     */

    matchedLocations.get(listPosition).setAddress(address);
```

```
        locationListAdapter.notifyItemChanged(listPosition);

        Log.d(LogTags.MatchFound_TAG, "updateAddress: address =
"+matchedLocations.get(listPosition).toString());



        updateCount++;
        if(updateCount>=matchedLocations.size()){

            locationsFetchFinishedFlag = true;

            updateCount = 0;

            if(homeLocationsFetchFinishedFlag)
                dataFetchFinishedUI();
  }
  }
}
```

TrackerSettingsActivity.java

```
package com.example.covid_19alertapp.activities;

import androidx.annotation.NonNull;
```

```
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.Manifest;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Switch;
import android.widget.Toast;

import com.example.covid_19alertapp.R;
import com.example.covid_19alertapp.extras.Constants;
import com.example.covid_19alertapp.extras.LocationFetch;
import com.example.covid_19alertapp.extras.LogTags;
import com.example.covid_19alertapp.extras.Notifications;
import com.example.covid_19alertapp.extras.Permissions;
import com.example.covid_19alertapp.services.BackgroundLocationTracker;
```

```
public class TrackerSettingsActivity extends AppCompatActivity {
/*
settings (currently only contains location on/off)
 */

    Button home_btn;
    Switch notification_switch;
    private static boolean switch_status;

    // for location permission
    private Permissions permissions;
    private static final String[] permissionStrings = {
            Manifest.permission.ACCESS_FINE_LOCATION,
            Manifest.permission.ACCESS_BACKGROUND_LOCATION,
            Manifest.permission.ACCESS_WIFI_STATE
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tracker_settings);
```

```
home_btn= findViewById(R.id.home_button_settings);

//start notification channel(do this is MainActivity
Notifications.createNotificationChannel(this);

notification_switch = findViewById(R.id.notification_switch);

home_btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        finish();
    }
});

notification_switch.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        save_preferences(notification_switch.isChecked());
        if(notification_switch.isChecked())
        {
            try {
```

```
LocationFetch.checkDeviceLocationSettings(TrackerSettingsActivity.this);
                if(LocationFetch.isLocationEnabled) {
                    // location is enabled
                    // start tracker service
                    Log.d(LogTags.Location_TAG, "onClick: location found enabled");

                    // start BackgroundLocationTracker
                    startTrackerService();
                }

                else{
                    // location is not enabled
                    Log.d(LogTags.Location_TAG, "onClick: location found disabled");

                    notification_switch.setChecked(false);
                    Toast.makeText(getApplicationContext(), "Turn on location or press again please", Toast.LENGTH_LONG)
                            .show();
                    save_preferences(false);
                }
            }catch (Exception e){
```

```
                    // set switch off
                    notification_switch.setChecked(false);

                    // set shared preferences false
                    save_preferences(false);

                    // most probable reason for error is permission not granted
                    promptPermissions();

                    Log.d(LogTags.TrackerSettings_TAG, "onClick: error starting background location service! permission taken?");
                }
            }
            else
            {
                try {
                    // stop location tracker
                    stopService(new Intent(getApplicationContext(),BackgroundLocationTracker.class));

                }catch (Exception e){
                    Log.d(LogTags.TrackerSettings_TAG, "onClick: error occured!");
```

```
                }
            }
        }
    });
    loadData();
    updateViews();
}

private void startTrackerService(){

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        startForegroundService(new Intent(getApplicationContext(), BackgroundLocationTracker.class));
        Log.d(LogTags.TrackerSettings_TAG, "onClick: newer version phones foreground service stared");
    } else
        startService(new Intent(getApplicationContext(), BackgroundLocationTracker.class));

}

private void promptPermissions() {
```

```
        permissions = new Permissions(this, permissionStrings,
Constants.PERMISSION_CODE);


        if(!permissions.checkPermissions())
            permissions.askPermissions();


    }



    public void save_preferences(boolean state)
    {
        SharedPreferences sharedPreferences =

getSharedPreferences(Constants.LOCATION_SETTINGS_SHARED_PREFER
ENCES, MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putBoolean(Constants.location_tracker_state,state);
        editor.apply();
    }
    public void loadData()
    {
        SharedPreferences sharedPreferences =

getSharedPreferences(Constants.LOCATION_SETTINGS_SHARED_PREFER
ENCES, MODE_PRIVATE);
```

```
        switch_status = sharedPreferences.getBoolean(Constants.location_tracker_state,false);
        updateViews();
    }

    public void updateViews()
    {
        notification_switch.setChecked(switch_status);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data){
        super.onActivityResult(requestCode, resultCode, data);

        switch (requestCode){

            case Constants.LOCATION_CHECK_CODE:
                // user input from the dialogbox showed after checkLocation()

                if(Activity.RESULT_OK == resultCode){
                    // user picked yes
                    Log.d(LogTags.Location_TAG, "onActivityResult: user picked yes. starting background location tracker");
```

```
                startTrackerService();

                // save settings preferences
                save_preferences(true);
                // set LocationFetch boolean
                LocationFetch.isLocationEnabled = true;

                //set the settings switch UI to true
                notification_switch.setChecked(true);
            }

            else if(Activity.RESULT_CANCELED == resultCode){
                // user picked no
                Log.d(LogTags.Location_TAG, "onActivityResult: user picked no. setting boolean and preference to false");

                save_preferences(false);
                LocationFetch.isLocationEnabled = false;
            }

            break;
```

```
        }


    }


    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
        //resolve unresolved permissions
 switch (requestCode case Constants.PERMISSION_CODE:
 try {
   this.permissions.resolvePermissions(permissions, grantResults);
                }catch (Exception e){
                    Log.d(LogTags.Permissions_TAG,  "onRequestPermissionsResult: "+e.getMessage());
                }
 break;
}}}
```

AddressPickerMapsActivity.java

```
 package com.example.covid_19alertapp.activities;
import androidx.annotation.NonNull;
import androidx.fragment.app.FragmentActivity;


import android.content.Context;
```

```
import android.content.Intent;
import android.location.Location;
import android.location.LocationManager;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.example.covid_19alertapp.R;
import com.example.covid_19alertapp.extras.AddressReceiver;
import com.example.covid_19alertapp.extras.Internet;
import com.example.covid_19alertapp.extras.LogTags;
import com.google.android.gms.common.api.Status;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
```

```
import com.google.android.libraries.places.api.Places;

import com.google.android.libraries.places.api.model.Place;

import com.google.android.libraries.places.api.model.TypeFilter;

import com.google.android.libraries.places.api.net.PlacesClient;

import com.google.android.libraries.places.widget.AutocompleteFragment;

import
com.google.android.libraries.places.widget.AutocompleteSupportFragment;

import
com.google.android.libraries.places.widget.listener.PlaceSelectionListener;


import java.util.Arrays;


public class AddressPickerMapsActivity extends FragmentActivity implements

        OnMapReadyCallback,

        GoogleMap.OnMyLocationButtonClickListener,

        GoogleMap.OnMyLocationClickListener,

        GoogleMap.OnMapLongClickListener {


    private GoogleMap mMap;

    private Button confirmButton;

    private Marker homeMarker = null;


    // home address location

    Location pickedLocation;
```

```
    // places api client
    PlacesClient placesClient;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_address_picker_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);


        if(!Internet.isInternetAvailable(this)) {
            // no internet, map not visible


            Toast.makeText(this, "No internet! Failed to load map.", Toast.LENGTH_LONG)
                    .show();


            TextView textView = findViewById(R.id.userHelperText);
            textView.setText(getString(R.string.map_no_internet_text));
```

```
        }

        initPlacesApi();

        confirmButton = findViewById(R.id.confirm_button);
    }

    private void initPlacesApi() {

        Places.initialize(getApplicationContext(),
getString(R.string.google_maps_key));
        placesClient = Places.createClient(this);

        // initialize fragment
        AutocompleteSupportFragment autocompleteFragment =
                (AutocompleteSupportFragment)
getSupportFragmentManager().findFragmentById(R.id.autocomplete_fragment
);

        // specify place type (find out more)
        autocompleteFragment
                .setPlaceFields(Arrays.asList(Place.Field.NAME,
Place.Field.LAT_LNG))
```

```
            .setCountries("BD")
            .setTypeFilter(TypeFilter.GEOCODE);

        // place selection listener
        autocompleteFragment.setOnPlaceSelectedListener(new
PlaceSelectionListener() {
            @Override
            public void onPlaceSelected(@NonNull Place place) {

                // move camera to place

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(place.getLatLng(
), 16.0f));

                Log.d(LogTags.Map_TAG, "onPlaceSelected: place selected =
"+place.getName()+" "+place.getLatLng());

            }

            @Override
            public void onError(@NonNull Status status) {

                Toast.makeText(AddressPickerMapsActivity.this, "please try again",
Toast.LENGTH_LONG)
                        .show();
```

```
            Log.d(LogTags.Map_TAG, "onError: place selection error = "+status.toString());


        }
    });



}



@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;


    // Add a marker in Dhaka and move the camera
    LatLng dhaka = new LatLng(23.7805733, 90.2792376);
    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(dhaka, 10.0f));


    // check if all are needed
    mMap.setMyLocationEnabled(true);
    mMap.getUiSettings().setMyLocationButtonEnabled(true);
    mMap.setOnMyLocationClickListener(this);
```

```
        mMap.setOnMyLocationButtonClickListener(this);
        mMap.setOnMapLongClickListener(this);

        Log.d(LogTags.Map_TAG, "onMapReady: map ready");
    }

    @Override
    public void onMapLongClick(LatLng latLng) {
        /*
        location selected by long press on map
        ask user to confirm
         */Log.d(LogTags.Map_TAG, "onMapLongClick: marker at = "+latLng.toString());
        pickedLocation = new Location(getLocalClassName());
        pickedLocation.setLatitude(latLng.latitude);
        pickedLocation.setLongitude(latLng.longitude);
        if(homeMarker!=null){
            homeMarker.remove();
        }
 homeMarker=mMap.addMarker(new MarkerOptions().position(latLng).title("Home"));
 Toast.makeText(
                this,
                "press 'Confirm' to confirm or select another",
```

```
            Toast.LENGTH_LONG
        ).show();
       confirmButton.setEnabled(true);
    }
@Override
    public boolean onMyLocationButtonClick() {
        /*
        notify user if location and/or wifi is inactive
         */

        String toastText = "";
        if(!wifiEnabled() && !locationEnabled())
            toastText = "Turn On both WiFi & Location";
        else if(!locationEnabled())
            toastText = "Turn On Location";
        else if(!wifiEnabled())
            toastText = "Turn On WiFi";

        if(!toastText.equals(""))
            Toast.makeText(this
                    , toastText + " to show your location"
                    , Toast.LENGTH_LONG)
                    .show();
```

```
        return false;
    }

    @Override
    public void onMyLocationClick(@NonNull Location location) {

        if(location.getAccuracy()>150)
            Toast.makeText(
                    this,
                    "Location Accuracy is LOW. press again please!"+location,
Toast.LENGTH_SHORT
            ).show();

    }public boolean wifiEnabled(){
        WifiManager wifi = (WifiManager) getApplicationContext()
                .getSystemService(Context.WIFI_SERVICE);
        return wifi.isWifiEnabled();
    }
 public boolean locationEnabled(){
        LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
        return
locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER) &&
```

```
locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDE
R);
   } public void confirmClicked(View view) {
       /*
       take this location and set it as home address
        */
 Log.d(LogTags.Map_TAG, "confirmClicked: location taken =
"+pickedLocation.toString());
Toast.makeText(this, "Your home location was saved!",
Toast.LENGTH_SHORT)
              .show();
// send data to parent activity
       Intent resultIntent = new Intent();
       resultIntent.putExtra("latitude-longitude",
              pickedLocation.getLatitude()+","+pickedLocation.getLongitude());
       setResult(RESULT_OK, resultIntent);
       finish();
 }}
```

DEBUGGING:

```
@if "%DEBUG%" == "" @echo off
@rem
##################################################################
##########
@rem
```

```
@rem Gradle startup script for Windows

@rem

@rem
################################################################################

@rem Set local scope for the variables with windows NT shell

if "%OS%"=="Windows_NT" setlocal

set DIRNAME=%~dp0

if "%DIRNAME%" == "" set DIRNAME=.

set APP_BASE_NAME=%~n0

set APP_HOME=%DIRNAME%


@rem Add default JVM options here. You can also use JAVA_OPTS and GRADLE_OPTS to pass JVM options to this script.

set DEFAULT_JVM_OPTS=

@rem Find java.exe

if defined JAVA_HOME goto findJavaFromJavaHome

set JAVA_EXE=java.exe

%JAVA_EXE% -version >NUL 2>&1

if "%ERRORLEVEL%" == "0" goto initecho.

echo ERROR: JAVA_HOME is not set and no 'java' command could be found in your PATH.

echo.

echo Please set the JAVA_HOME variable in your environment to match the
```

```
echo location of your Java installation.

goto fail:findJavaFromJavaHome

set JAVA_HOME=%JAVA_HOME:"=%

set JAVA_EXE=%JAVA_HOME%/bin/java.exe


if exist "%JAVA_EXE%" goto init


echo.

echo ERROR: JAVA_HOME is set to an invalid directory: %JAVA_HOME%

echo.

echo Please set the JAVA_HOME variable in your environment to match the

echo location of your Java installation.

goto fail:init

@rem Get command-line arguments, handling Windows variants

if not "%OS%" == "Windows_NT" goto win9xME_args:win9xME_args

@rem Slurp the command line arguments.

set CMD_LINE_ARGS=

set _SKIP=2:win9xME_args_slurp

if "x%~1" == "x" goto execute

set CMD_LINE_ARGS=%*:execute

@rem Setup the command line

set CLASSPATH=%APP_HOME%\gradle\wrapper\gradle-wrapper.jar

@rem Execute Gradle
```

"%JAVA_EXE%" %DEFAULT_JVM_OPTS% %JAVA_OPTS% %GRADLE_OPTS% "-Dorg.gradle.appname=%APP_BASE_NAME%" -classpath "%CLASSPATH%" org.gradle.wrapper.GradleWrapperMain %CMD_LINE_ARGS%:end

@rem End local scope for the variables with windows NT shell

if "%ERRORLEVEL%"=="0" goto mainEnd

:fail

rem Set variable GRADLE_EXIT_CONSOLE if you need the _script_ return code instead of

rem the _cmd.exe /c_ return code!

if not "" == "%GRADLE_EXIT_CONSOLE%" exit 1

exit /b 1:mainEnd

if "%OS%"=="Windows_NT" endlocal:omega

GitHubLink: **IBM-Project-11907-1659356090**